UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Una app móvil colaborativa para la creación de diagramas de casos de uso

Autor: Federico Baras de Castro
Tutor: Juan De Lara Jaramillo

Junio 2021

Una app móvil colaborativa para la creación de diagramas de casos de uso

AUTOR: Federico Baras de Castro
TUTOR: Juan De Lara Jaramillo

Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2021

Resumen

La finalidad de este Trabajo de Fin de Grado es la creación de una aplicación colaborativa y multiplataforma para la creación de diagramas de casos de uso.

El objetivo es tener una aplicación móvil en la cual se realice el modelado con la posibilidad de mostrarlo en una aplicación web.

Para cumplir con estos requisitos, ha sido necesario la creación de tres aplicaciones. La aplicación móvil, se ha desarrollado en Java para la plataforma Android, y es la responsable de permitir el registro, acceso, creación y modificación de los diagramas. La aplicación web, tendrá dos tipos de acceso. Un acceso para administrador que permitirá la gestión de los usuarios y el envío de las notificaciones y otro acceso para usuarios que permitirá mostrar los diagramas diseñados en la aplicación móvil en una pantalla mayor. Este desarrollo se realizará en Angular con Bootstrap. Por ultimo, para conectar ambas plataformas con el modelo de datos, se creará un servidor en Java, utilizando para ello la plataforma Spring.

Palabras clave

Diagramas de casos de uso, modelado colaborativo, desarrollo móvil, Android, Spring, Angular, Bootstrap.

Abstract

The purpose of this Final Degree Project is the creation of a collaborative and multiplatform application for the creation of use case diagrams.

The objective is to have a mobile application in which the modeling is carried out with the possibility of displaying it in a web application.

To meet these requirements, it was necessary to create three applications. The mobile application has been developed in Java for the Android platform, and is responsible for allowing the registration, access, creation and modification of the diagrams. The web application will have two types of access. An administrator access that will allow the management of users and the sending of notifications and an access for users that will allow the diagrams designed in the mobile application to be displayed on a larger screen. This development will be done in Angular with Bootstrap. Finally, to connect both platforms with the data model, a server will be created in Java, using the Spring platform.

Keywords

Use case diagrams, collaborative modeling, mobile development, Android, Spring, Angular, Bootstrap.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Motivación

El modelado de requisitos es una actividad colaborativa, que implica la coordinación entre ingenieros del software y expertos en el dominio. Para permitir un proceso flexible, este TFG plantea la creación de una aplicación móvil colaborativa para la creación de diagramas de casos de uso. La aplicación será colaborativa, permitiendo a varios usuarios la creación conjunta de diagramas de casos de uso a través de un servidor central. Este esquema de colaboración síncrona permitirá escenarios donde los ingenieros puedan modelar en sus tabletas o móviles, y se muestre en una pantalla de mayores dimensiones el modelo compartido.

1.2 Objetivos

El objetivo principal de este TFG es el análisis y desarrollo de una aplicación móvil colaborativa para la creación de diagramas de casos de uso. Este objetivo está desglosado en los siguientes apartados:

- Desarrollo de un servidor, accesible mediante API para la gestión de usuarios y la lógica de negocio de la gestión de los diagramas.
- Desarrollo de una aplicación web para la visualización de los diagramas.
- Desarrollo de una aplicación móvil para el modelado de los diagramas.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Estado del arte.** En esta sección se analizará un conjunto de herramientas existentes similares al proyecto que nos ocupa y hablaremos del conjunto de herramientas y tecnologías que se han utilizado para llevar a cabo el proyecto.
- **Análisis y diseño.** En esta sección se describirán los requisitos funcionales y no funcionales para el desarrollo de la aplicación.
- **Desarrollo.** Aquí se expondrá una compilación de las tareas llevadas a cabo en cada fase del desarrollo.
- **Integración y pruebas.** Integración de los distintos módulos y pruebas.
- **Conclusiones y trabajo futuro.** En esta sección se evaluará el trabajo desarrollado, se extraerán conclusiones y se hará un compendio de posibles mejoras a llevar a cabo en el futuro.

2 Estado del arte

Esta sección se compone de dos apartados. En el primero se hace una revisión de las herramientas existentes orientadas al modelado cooperativo y en el segundo se lista el conjunto de herramientas y tecnologías utilizadas durante el desarrollo.

2.1 Herramientas para modelado

En este apartado se presentan varias herramientas de modelado de propósito similar al propuesto por este TFG.

A Collaborative Multi-Touch UML Design Tool [1]: es una herramienta para el modelado de diagramas UML. Esta orientada al modelado colaborativo en pantallas grandes, con varios usuarios trabajando en la misma. Tiene como aspecto significativo el uso de gestos para la creación y manipulación de diagramas. Por otro lado, el trabajo se complica cuando hay mas de dos usuarios, cuando se trabaja sobre el mismo elemento o a la hora de introducir texto. También hay que tener en cuenta que no es una herramienta online ni multiplataforma.

Cel – Modeling Everywhere [2]: es una herramienta de modelado para tabletas. Para conseguir una herramienta bien estructurada y ligera, basa el modelado en un sistema que almacena las entidades y relaciones del diagrama en las celdas de una matriz. Para lidiar con pantallas pequeñas, permite ampliar o reducir la imagen mostrada en la interfaz. A diferencia de la propuesta de este trabajo, no permite el modelado colaborativo y no es multiplataforma.

FlexiSketch [3]: es una herramienta de modelado para Android. Permite definir los diagramas mediante gestos y exportar los diagramas como imágenes. Es bastante ligera e intuitiva pero tampoco permite el modelado colaborativo y no es multiplataforma.

Cacoo [2]: potente software para el modelado online. Permite el modelado de múltiples elementos: diagramas de flujo, maquetas, diagramas de red, presentaciones, diagramas de base de datos, diagramas de Gantt, etc. Está muy enfocada al trabajo colaborativo, permitiendo la edición simultanea, el seguimiento de cambios y dispone de chat de texto y video. Tiene muy buena capacidad de integración con un gran abanico de plataformas. Aunque es una herramienta muy completa, esta basada en la web y no dispone de una aplicación para dispositivos móviles, al contrario que el trabajo propuesto en este TFG.

UMLetino [5]: es una herramienta online gratuita para el modelado de diagramas UML. Está basada en UMLet [6], una herramienta gratuita de código abierto para diseñar diagramas UML. Tiene una interfaz sencilla y ligera donde permite crear diagramas

seleccionando los elementos y arrastrándolos hasta el espacio de trabajo. Permite guardar los elementos en varios formatos y tiene integración con Dropbox. Al igual que la herramienta anteriormente, esta enfocada a la web y, además, no es posible el trabajo colaborativo simultaneo.

Visual Paradigm Online [7]: versión online de la reconocida herramienta de escritorio para el modelado de diagramas. Permite la colaboración simultanea, tiene un amplio conjunto de plantillas para escoger, filtros, efectos, iconos, ilustraciones, etc. Nuevamente, nos encontramos con una herramienta muy potente, pero que no dispone de versión para dispositivos móviles.

Lucidchart [8]: es una herramienta de modelado web que permite a los usuarios colaborar en tiempo real para la realización de todo tipo de diagramas. Además del potente editor web y a diferencia del resto de herramientas analizadas, cuenta con una aplicación móvil donde ver y editar los diagramas asociados al usuario. Aunque es una herramienta muy completa, la aplicación móvil no cuenta con el seguimiento de cambios y el sistema de notificaciones que si tiene la aplicación desarrollada en este TFG.

2.2 Tecnologías y herramientas

En esta sección describiremos el conjunto de herramientas y tecnologías que se han utilizado para llevar a cabo el proyecto. Dado que el proyecto se divide en tres aplicaciones diferenciadas, agruparemos estas tecnologías conforme a la aplicación en la que se le da uso. Las aplicaciones son:

- Aplicación servidora
- Aplicación web
- Aplicación móvil.

En el primer apartado, se describirán las herramientas y tecnologías comunes a todo el proyecto.

2.2.1 Tecnologías y herramientas comunes

- **REST**. Tipo de arquitectura web utilizado como interfaz entre sistemas que se ejecuta sobre HTTP. En este caso permitirá la comunicación entre la aplicación servidora y las aplicaciones web y móvil. Como tipo de datos se utilizará JSON.

- **GIT**. Software de control de versiones. Su propósito es registrar los cambios en los archivos del proyecto, incluyendo la modificación colaborativa de los mismos.

- **GitLab**. Servicio web de control de versiones y desarrollo de software colaborativo basado en Git. Se ha utilizado un repositorio para cada una de las tres aplicaciones que componen el proyecto [9].

- **Apache Tomcat**. Contenedor de *servlets*. Proporciona un servidor web HTTP en el cual ejecutar código Java [10].

- **JSON**. Formato de texto ligero para el intercambio de datos. Es la principal alternativa al XML como tipo de datos a utilizar por la REST API y la opción a utilizar en el proyecto [11].

2.2.2 Tecnologías para la aplicación servidora

- **Java**. Lenguaje de programación de alto nivel, basado en clases, orientado a objetos y multiplataforma [12].

- **Spring**. Framework para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java [13].

- **JPA (Java Persistence API)**. Framework de Java, que forma parte de Spring y que maneja datos relacionales en aplicaciones bajo plataforma Java. Se ha elegido este sistema para la persistencia en BBDD porque aporta mucha simplicidad al proceso en casos de bases de datos sin excesiva complejidad.

- **MySQL**. Sistema de gestión de bases de datos relacionales de código abierto [14].

- **JWT (JSON Web Token)**. Estándar basado en JSON para la creación de *tokens* de acceso utilizado principalmente en autenticación [15].

- **Lombok**. Librería java que resulta de mucha utilidad porque simplifica mucho la creación de clases ya que nos evita tener que introducir en cada una los métodos "getters", "setters", constructores, o métodos como "equals" y "hashCode" [16].

2.2.3 Herramientas para la aplicación servidora

- **Intellij IDEA**. Entorno de desarrollo integrado (IDE) para el desarrollo de programas informáticos. Durante este desarrollo, se utilizará la versión Ultimate. Como ventajas al utilizar este IDE están el control de versiones integrado, el análisis de código y la herramienta de gestión de bases de datos [17].

- **Maven.** Herramienta software para la gestión y construcción de proyectos Java. Se ha elegido Maven por encima de otras alternativas como Gradle, por contar con un repositorio [18].

- **DBeaver.** Aplicación de software cliente de SQL y herramienta de administración de bases de datos. Se elige este por ser de código abierto y estar disponible para varios sistemas operativos [19].

- **Postman.** Herramienta para realizar peticiones HTTP. De gran utilidad para probar la API [20].

- **HTTPPie.** Cliente HTTP de línea de comandos. Similar a Postman, pero usando el terminal [21].

2.2.4 Tecnologías para la aplicación web

- **Angular.** Framework para aplicaciones web desarrollado en TypeScript, de código abierto, que se utiliza para crear y mantener aplicaciones web [22].

- **HTML.** Lenguaje de marcado para la elaboración de paginas web.

- **CSS.** Lenguaje de diseño grafico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado.

- **JavaScript.** Lenguaje de programación interpretado utilizado para el desarrollo de paginas web, principalmente en el lado cliente.

- **JQuery.** Framework Javascript de código abierto que simplifica la manera de interactuar con los documentos HTML.

- **BootStrap.** Framework CSS de código abierto para diseño de aplicaciones web [23].

- **FontAwesome**. Es un *toolkit* de fuentes e iconos basado en CSS y Lens [24].

- **Ngx-Graph**. Es una librería para visualización de grafos para Angular [25].

2.2.5 Herramientas para la aplicación web

- **Visual Studio Code.** Editor de código con soporte para depuración, control de versiones refactorización, etc. Es muy ligero, personalizable y de código abierto.

- **NodeJs.** Entorno de programación usado principalmente en el lado del servidor. Se utilizará mara montar el servidor web en el cual correrá la aplicación de Angular.

- **Google Chrome.** Navegador web. Se escoge este navegador principalmente por las herramientas integradas de cara a la depuración.

- **NPM.** Node Package Manager. Sistema de gestión de paquetes de Node.Js. Se ejecuta desde línea de comandos y maneja las dependencias de la aplicación.

2.2.6 Tecnologías para la aplicación móvil

- **Android**. Sistema operativo móvil basado en Linux. Es el sistema operativo más utilizado y es por eso el elegido para el desarrollo de la aplicación móvil.

- **Retrofit**. Librería para Android y Java que actúa como cliente HTTP. Simplifica en gran medida la comunicación con la API REST por medio de las anotaciones, la posibilidad de enviar peticiones asíncronas y la capacidad de utilizar múltiples conversores para JSON y SML [26].

2.2.7 Herramientas para la aplicación móvil

- **Android Studio**. Entorno de desarrollo oficial para la plataforma Android. Esa basado en IntelliJ IDEA, pero diseñado específicamente para el desarrollo en Android.

3 Análisis y diseño

Esta sección se divide en los bloques Análisis y Diseño. En el primero, se presentarán los requisitos recogidos en la fase de análisis de requisitos con un análisis estructurado y otro orientado a objetos con diagramas de uso. En la parte de diseño, se presentará la arquitectura del sistema y el modelo de la base de datos.

3.1 Análisis

Antes de proceder con el diseño, tiene lugar la captura de requisitos, una de las fases más importantes en la definición de los requisitos. Dividiremos los requisitos en funcionales y no funcionales. Cada requisito constará de descripción, entrada, proceso y salida. Una vez listados los requisitos, se representarán con diagramas de uso.

3.1.1 Requisitos Funcionales

A continuación, se exponen los requisitos funcionales a desarrollar durante el ciclo de vida del proyecto, separados para cada uno de los dos actores implicados y también según la plataforma utilizada.

Requisitos comunes para aplicación móvil y web

- **(RF1) Registro de un usuario**. Permite el registro de un usuario.

  Entrada: El usuario accede en la aplicación web o la aplicación móvil al apartado de registro donde introduce el nombre de usuario, la contraseña y el correo electrónico y presiona el botón de registro.

  Proceso: El sistema comprueba que el usuario y el correo electrónico no estén en uso y que tanto el usuario como la contraseña tengan un formato correcto.

  Salida: Si los datos son correctos, el sistema redirige al usuario a la pagina de bienvenida y si no son correctos, se informa al usuario con un mensaje.

- **(RF2) Inicio de sesión**. Otorga el acceso del usuario a la aplicación.

  Entrada: El usuario introduce el nombre de usuario y la contraseña.

Proceso: El sistema comprueba que el usuario existe en el sistema y que la contraseña coincide con la almacenada en el sistema para ese usuario.

Salida: Si los datos son correctos, el sistema redirige al usuario a la pagina de bienvenida y si no son correctos, se informa al usuario con un mensaje de error.

Aplicación web

- **(RF3) Envío de notificación**. En el caso de que el usuario que ha iniciado sesión en el sistema sea administrador, tiene acceso al panel de notificaciones, donde podrá enviar una notificación a cualquier usuario del sistema.

  Entrada: El administrador introduce el mensaje que compone la notificación y selecciona de un campo desplegable el usuario que la recibirá, tras lo cual presiona el botón de enviar.

  Proceso: El sistema comprueba que se ha seleccionado algún elemento en el despegable y que el campo de texto no está vacío.

  Salida: Si los datos son correctos, la notificación se incorpora al sistema y el usuario es informado de ello en un mensaje. Si la comprobación no es satisfactoria, se le mostrará al usuario un mensaje de error.

- **(RF4) Edición de un usuario**. Si el usuario es administrador, tiene acceso al panel de gestión de usuarios, donde podrá editar los usuarios.

  Entrada: El administrador accede a la web de la aplicación en el apartado de gestión de usuarios donde selecciona uno, accede a la edición, modifica los datos y selecciona guardar.

  Proceso: El sistema comprueba si los nuevos datos son correctos, esto es, si el usuario y el correo no están en uso y tienen el formato correcto.

  Salida: Si los datos son correctos, se modifica el usuario en el sistema. Si los datos son erróneos o se produce algún error en la llamada, se mostrará mediante un mensaje de error.

- **(RF5) Eliminación de un usuario**. Si el usuario es administrador, tiene acceso al panel de gestión de usuarios, donde podrá eliminar usuarios.

Entrada: El administrador accede a la web de la aplicación en el apartado de gestión de usuarios donde se muestra una lista con los usuarios del sistema. Se selecciona uno y se presiona el botón de eliminar.

Proceso: El sistema comprueba que el usuario existe en el sistema.

Salida: Si los datos son correctos, se elimina el usuario del sistema. Si los datos son erróneos o se produce algún error en la llamada, se mostrará mediante un mensaje de error

- **(RF6) Agregar un usuario**. Si el usuario es administrador, tiene acceso al panel de gestión de usuarios, donde podrá agregar usuarios.

  Entrada: El administrador accede a la web de la aplicación en el apartado de gestión de usuarios donde se muestra una lista con los usuarios del sistema. Presiona el botón para agregar un usuario, rellena los datos y presiona el botón de guardar.

  Proceso: El sistema comprueba que el usuario no existe en el sistema y que los campos son válidos.

  Salida: Si los datos son correctos, se agrega el usuario al sistema. Si los datos son erróneos o se produce algún error en la llamada, se mostrará mediante un mensaje de error

Aplicación móvil

- **(RF7) Agregar contacto**. El usuario tiene la posibilidad de agregar un contacto si conoce el email del usuario que quiere agregar.

  Entrada: El usuario accede a la pestaña de contactos y presiona el botón de añadir, se despliega una ventana emergente que da la opción de introducir un texto, tras lo cual, se presión el botón de enviar.

  Proceso: El sistema comprueba que el email está registrado en el sistema.

  Salida: Si el email existe, se crea el contacto, pero este no aparecerá en la lista de contactos hasta que el usuario al que corresponde el email haya aceptado la solicitud de amistad. Si el campo esta vacío o el usuario ya existe en la lista de contactos, se mostrará un mensaje de error.

- **(RF8) Eliminar contacto**. El usuario tiene la posibilidad de eliminar un contacto de su lista de contactos.

  Entrada: El usuario accede a la pestaña de contactos, selección un contacto de la lista y presiona el botón de eliminar.

  Proceso: El sistema comprueba que hay un contacto seleccionado en la lista y comprueba que el usuario existe en el sistema.

  Salida: Si las condiciones se cumplen, el usuario se elimina de la lista de contactos y se muestra un mensaje de confirmación. Si no hay un contacto seleccionado, el botón de eliminar estará deshabilitado y si hay algún error en la conexión, se mostrará un mensaje de error.

- **(RF9) Visualizar contacto**. El usuario tiene la posibilidad de ver los detalles de sus contactos, esto es, el nombre de usuario y el email.

  Entrada: El usuario accede a la pestaña de contactos, selección un contacto de la lista y presiona el botón de visualizar.

  Proceso: El sistema comprueba que hay un contacto seleccionado en la lista y comprueba que el usuario existe en el sistema.

  Salida: Si las condiciones se cumplen, se abre una ventana emergente donde se muestran los detalles del contacto. Si no hay un contacto seleccionado, el botón de visualizar estará deshabilitado y si hay algún error en la conexión, se mostrará un mensaje de error.

- **(RF10) Visualizar notificación**. El usuario tiene la posibilidad de ver los detalles de sus notificaciones.

  Entrada: El usuario accede a la pestaña de notificaciones, selección una notificación de la lista y presiona el botón de visualizar.

  Proceso: El sistema comprueba que hay una notificación seleccionada en la lista.

  Salida: Si las condiciones se cumplen, se abre una nueva actividad donde se mostrará la notificación completa. Si no hay una notificación seleccionada, el botón de visualizar estará deshabilitado y si hay algún error en la conexión, se mostrará un mensaje de error.

- **(RF11) Eliminar notificación**. El usuario tiene la posibilidad de eliminar notificaciones.

  Entrada: El usuario accede a la pestaña de notificaciones, selección una notificación de la lista y presiona el botón de eliminar.

  Proceso: El sistema comprueba que hay una notificación seleccionada en la lista.

  Salida: Si las condiciones se cumplen, la notificación se eliminará del sistema. Si no hay una notificación seleccionada, el botón de visualizar estará deshabilitado y si hay algún error en la conexión, se mostrará un mensaje de error.

- **(RF12) Crear diagrama**. El usuario tiene la posibilidad de crear diagramas.

  Entrada: El usuario accede a la pestaña de diagramas, y presiona el botón de añadir. Aparece una ventana emergente donde se permite introducir un texto y presiona el botón de confirmación.

  Proceso: El sistema comprueba que el campo de texto no está vacío y que el nombre de diagrama no existe.

  Salida: Si las condiciones se cumplen, se crea el diagrama en el sistema, se añade al usuario como participante de este y se añade el primer cambio al diagrama, que sería la creación de este. Si el campo está vacío, el nombre está repetido o hay algún error en la conexión, se mostrará un mensaje de error.

- **(RF13) Eliminar diagrama**. El usuario tiene la posibilidad de eliminar diagramas en los que conste como participante.

  Entrada: El usuario accede a la pestaña de diagramas, y presiona el botón de eliminar. Aparece una ventana emergente con una lista de los diagramas en los que participa, selección uno de ellos y presiona el botón de confirmación.

  Proceso: El sistema comprueba que el diagrama existe en el sistema.

  Salida: Si las condiciones se cumplen, se elimina el diagrama en el sistema junto con todos sus elementos asociados. Si no existe o hay algún error en la conexión, se mostrará un mensaje de error.

- **(RF14) Agregar participante al diagrama**. Una vez creado el diagrama, el usuario tiene la posibilidad de agregar participantes.

Entrada: El usuario accede a la pestaña de diagramas, presiona la tarjeta del diagrama al que quiera acceder y en la barra de herramientas, selecciona el icono de participantes. Dentro de la sección, selecciona un contacto de la lista y presiona el botón de añadir.

Proceso: El sistema comprueba que el usuario no exista en la lista de participantes.

Salida: Si las condiciones se cumplen, el usuario se agrega a la lista de participantes, se añade el cambio a la lista de cambios y a partir de ese momento, este otro usuario tendrá también acceso a la visualización y edición del diagrama. Si el usuario ya existe en la lista de participantes o hay algún error en la conexión, se mostrará un mensaje de error.

- **(RF15) Visualizar lista de cambios del diagrama**. Una vez creado el diagrama, el usuario tiene la posibilidad de visualizar la lista de cambios.

  Entrada: El usuario accede a la pestaña de diagramas, presiona la tarjeta del diagrama al que quiera acceder y en la barra de herramientas, selecciona el icono de cambios.

  Proceso: El sistema recupera la lista de cambios.

  Salida: El sistema muestra la lista o si hay algún error en la conexión, se mostrará un mensaje de error.

- **(RF16) Añadir actor al diagrama**. Tras crear el diagrama, el usuario puede añadir actores al mismo.

  Entrada: El usuario accede a la pestaña de diagramas, presiona la tarjeta del diagrama al que quiera acceder y selecciona el icono para agregar actor. Se muestra una ventana emergente donde introduce el nombre del actor y presiona el botón de confirmación.

  Proceso: El sistema comprueba que el campo de texto no está vacío y que el nombre no esté repetido.

  Salida: Si las condiciones se cumplen, se añade el actor al diagrama, se muestra en la lista de actores de este, se añade el cambio a la lista de cambios y se informa con un mensaje. Si el campo está vacío, el nombre está repetido o hay algún error en la conexión, se mostrará un mensaje de error.

- **(RF17) Eliminar actor del diagrama**. Tras crear el diagrama, el usuario puede eliminar actores de este.

  Entrada: El usuario accede a la pestaña de diagramas, presiona la tarjeta del diagrama al que quiera acceder y selecciona el icono para eliminar actor. Se muestra una ventana emergente donde selecciona el nombre del actor y presiona el botón de confirmación.

  Proceso: El sistema comprueba que el actor existe en el diagrama.

  Salida: Si la condición se cumple, se elimina el actor del diagrama, se actualiza la lista de actores de este, se añade el cambio a la lista de cambios y se informa con un mensaje. Si hay algún error en la conexión, se mostrará un mensaje de error.

- **(RF18) Añadir caso de uso al diagrama**. Tras crear el diagrama, el usuario puede añadir casos de uso a este.

  Entrada: El usuario accede a la pestaña de diagramas, presiona la tarjeta del diagrama al que quiera acceder y selecciona el icono para agregar un caso de uso. Se muestra una ventana emergente donde introduce el nombre del caso de uso y presiona el botón de confirmación.

  Proceso: El sistema comprueba que el campo de texto no está vacío y que no esté repetido.

  Salida: Si la condición se cumple, se añade el caso de uso al diagrama, se muestra en la lista de casos de uso de este, se añade el cambio a la lista de cambios y se informa con un mensaje. Si el campo está vacío, el nombre está repetido o hay algún error en la conexión, se mostrará un mensaje de error.

- **(RF19) Eliminar caso de uso del diagrama**. Tras crear el diagrama y guardar algún caso de uso, el usuario puede eliminar casos de usos del diagrama.

  Entrada: El usuario accede a la pestaña de diagramas, presiona la tarjeta del diagrama al que quiera acceder y selecciona el icono para eliminar casos de uso. Se muestra una ventana emergente donde selecciona el nombre del caso de uso y presiona el botón de confirmación.

  Proceso: El sistema comprueba que el caso de uso existe.

  Salida: Si la condición se cumple, se elimina el caso de uso del diagrama, se actualiza la lista de casos de uso de este, se añade el cambio a la lista de cambios y

se informa con un mensaje. Si el campo está vacío o hay algún error en la conexión, se mostrará un mensaje de error.

- **(RF20) Añadir relación**. Tras crear el diagrama, el usuario puede añadir relaciones.

  Entrada: El usuario accede a la pestaña de diagramas, presiona la tarjeta del diagrama al que quiera acceder y selecciona el icono para agregar relaciones. Se accede a una nueva actividad donde selecciona una entidad origen, una entidad destino, el tipo de relación y presiona el botón para agregar relación.

  Proceso: El sistema comprueba que hay elementos seleccionados en las tres listas.

  Salida: Si las condiciones se cumplen, se añade la relación al diagrama, se añade el cambio a la lista de cambios y se vuelve a la pagina de detalle del diagrama. Si no hay campos seleccionados en las tres listas, la relación no es admisible o hay algún error en la conexión, se mostrará un mensaje de error.

- **(RF21) Eliminar relación**. Tras crear el diagrama y disponer de relaciones, el usuario puede eliminarlas.

  Entrada: El usuario accede a la pestaña de diagramas, presiona la tarjeta del diagrama al que quiera acceder y selecciona el icono para eliminar relaciones. Se accede a una nueva actividad donde selecciona una relación de la lista de relaciones existentes y presiona el botón para eliminar relación.

  Proceso: El sistema comprueba que hay elementos seleccionados en la lista.

  Salida: Si la condición se cumple, se elimina la relación del diagrama, se añade el cambio a la lista de cambios y se vuelve a la pagina de detalle del diagrama. Si no hay campos seleccionados en la lista o hay algún error en la conexión, se mostrará un mensaje de error.

3.1.2 Requisitos no funcionales

- **(RNF1) Requisitos de seguridad**

  (RNF 1.1) La API estará protegida utilizando autenticación para evitar accesos no autorizados.

  (RNF 1.2) Existirán dos roles diferentes, usuario y administrador, para evitar que el usuario acceda a opciones avanzadas.

(RNF 1.3) Las contraseñas se encriptarán con un algoritmo, para evitar en lo posible brechas de seguridad en el acceso.

- **(RNF2) Requisitos de extensibilidad**

  (RNF 2.1) El proyecto seguirá las convecciones de software para asegurar que se pueda agregar nueva funcionalidad de manera sencilla.

3.1.3 Diagramas de casos de uso

A partir de los requisitos recogidos, se elaboran los diagramas de casos de uso para los dos roles existentes en el sistema.

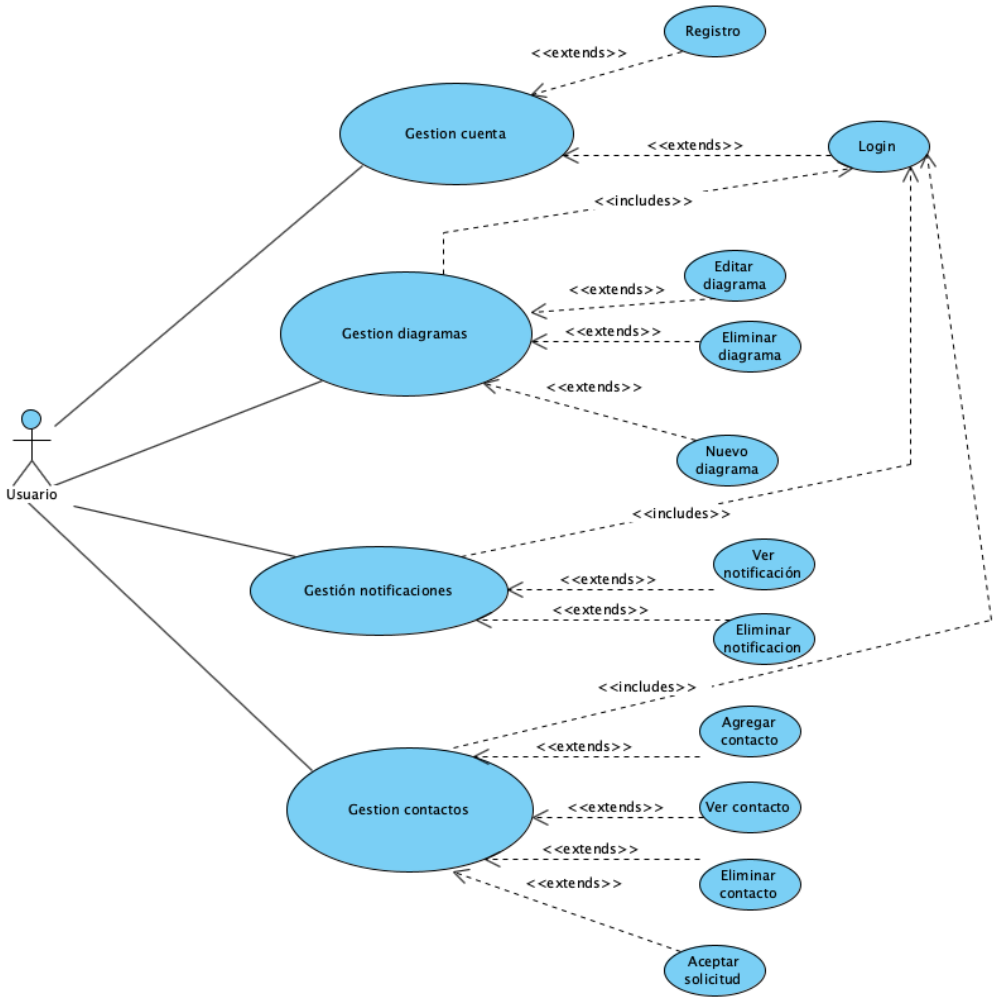

Figura 1: Diagrama Usuario General.

En este diagrama, podemos observar los principales casos de uso a los que tiene acceso el usuario, con un desglose de los casos de uso que los forman. Dentro de la gestión de la cuenta, se incluye el registro en el sistema y el inicio de sesión. Las otras actividades de gestión, para los diagramas, notificaciones y contactos, incluirán el acceso en el sistema.

La gestión de notificaciones incluye la posibilidad de ver la notificación y de eliminarla, mientras que la gestión de contactos incluye ver un contacto, agregar el contacto por medio de un email, eliminarlo y aceptar una solicitud de amistad.

En cuanto a la gestión del diagrama, tenemos la posibilidad de agregar un diagrama, editarlo y eliminarlo. La edición del diagrama, que es un caso de uso mas complejo, se muestra a continuación.

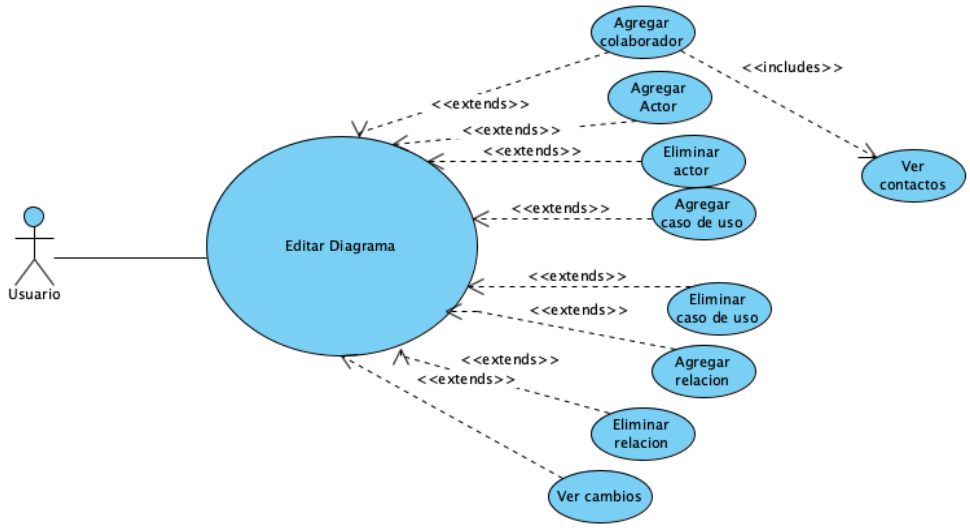

Figura 2: Diagrama usuario. Editar diagrama.

Editar el diagrama es el caso de uso principal de la aplicación. Se desglosa en una serie de casos de usos agrupados en dos funciones. La primera es la edición de los componentes asociados al modelado del diagrama. Estos casos de uso incluyen la agregación y la eliminación de actores, relaciones y casos de usos. Luego están una serie casos de usos, que están asociados al diagrama, pero no son parte del diseño. El caso de uso para agregar un colaborador permitirá buscar en nuestra lista de contactos para agregar un nuevo usuario a la lista de gestores del diagrama. El caso de uso para ver cambios permite hacer un seguimiento de los cambios realizados sobre el diagrama por fecha y usuario.

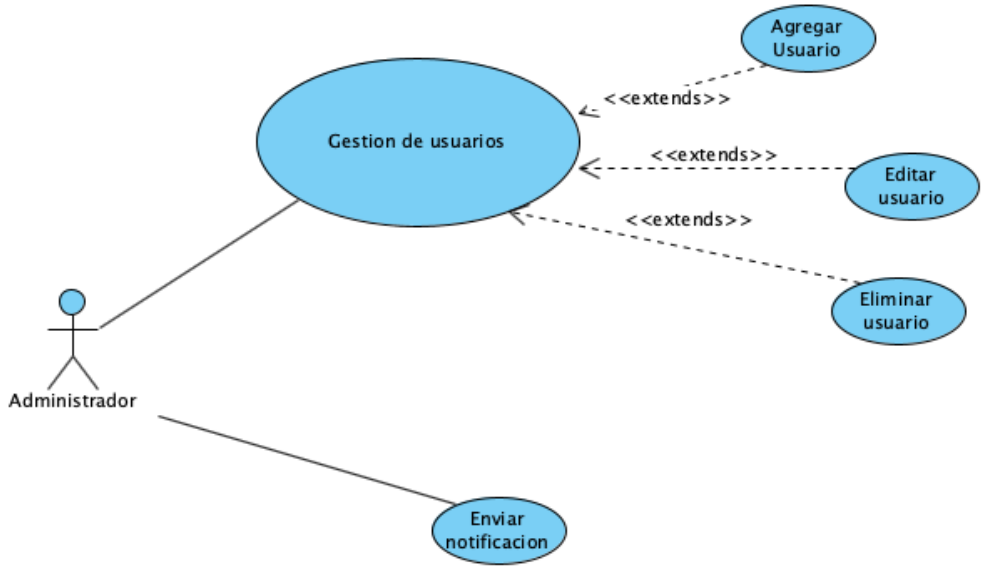

Figura 3: Diagrama administrador.

El usuario administrador, tiene acceso a la gestión de los usuarios y al envío de notificaciones. La gestión de usuarios incluye la creación, edición y eliminación. Estos usos, están restringidos a la aplicación web.

3.2 Diseño

En este apartado se presenta mediante diagramas la arquitectura del sistema y el modelo de base de datos.

3.2.1 Arquitectura del sistema

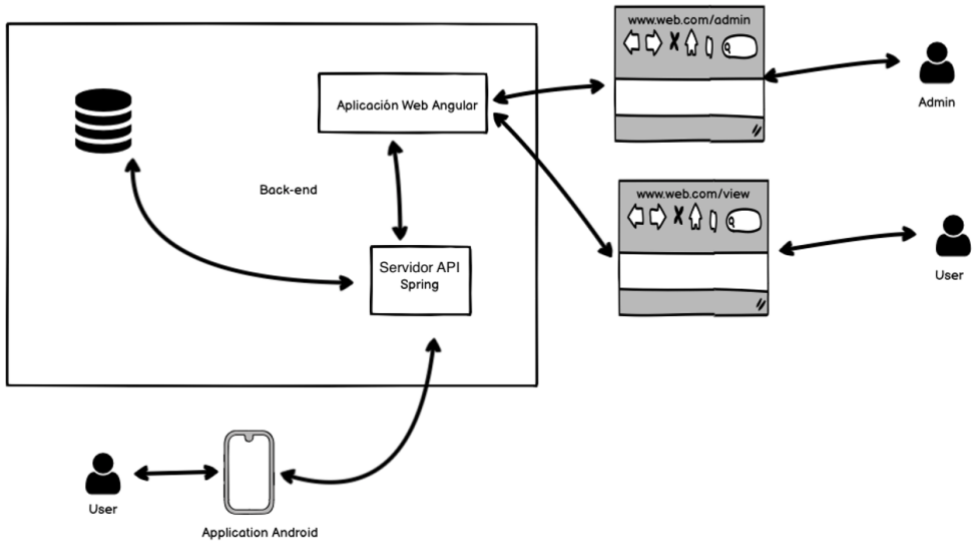

Figura 4: Arquitectura del sistema.

En la figura podemos observar la composición del sistema. Por un lado, tenemos la aplicación móvil, para Android, que es la herramienta que va a utilizar el usuario para el modelado. Los diagramas modelados, será posible visualizarlos en la web, gracias a la aplicación web Angular alojada en el servidor web. Tanto la aplicación móvil como la aplicación web, consumirán los servicios REST del servidor Java creado con Spring. Este servidor será el encargado de la persistencia de los datos en la base de datos.

El proyecto se divide en tres aplicaciones:

- Aplicación servidora. Desarrollada en Java con Spring. La BBDD será MySQL.
- Aplicación web. Desarrollada en Angular con BootStrap.
- Aplicación móvil. Desarrollada en Java para la plataforma Android.

3.2.2 Modelo de datos

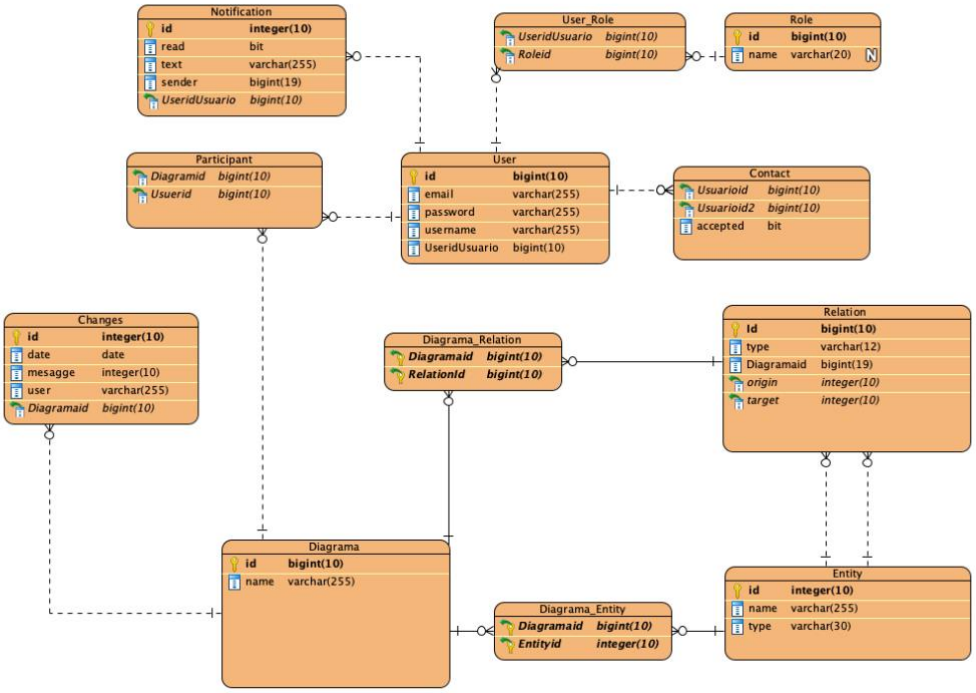

Figura 5: Diagrama de la base de datos.

El modelo de datos utilizado en el proyecto se divide en dos grupos. Por un lado, están las entidades que tienen en común los usuarios y su gestión, y por otro lado las entidades relacionadas con el modelado de los diagramas y sus componentes.

Así, tenemos por una parte la entidad **User**, que representa al usuario de la aplicación que se da de alta en el sistema y que tendrá los atributos necesarios para su autenticación y personalización. Para la gestión del acceso de los usuarios, como viene especificado en el requisito **RNF1.2**, se ha creado la entidad **Role**, que se encargará de identificar el rol del usuario, para poder asignar los usos de la aplicación dependiendo de si este es un usuario normal o un usuario administrador. Esta entidad, está relacionada con el usuario por medio de la tabla asociada **User_Role**. Luego tendremos la tabla asociada **Contact**, que establece las relaciones entre usuarios y tiene un atributo que marca la situación de la solicitud de amistad. Y por último tenemos **Notification**, que es la entidad que define las notificaciones asociadas al usuario.

En cuanto a la gestión del diagrama, tenemos **Diagram**, que está relacionada con los usuarios a través de la tabla intermedia **Participant.** El diagrama, se compondrá de relaciones, definido en la entidad **Relation** y entidades, definido en **Entity**. Las entidades,

tendrán un campo tipo para diferenciar entre actores y usos. Las relaciones entre entidades vendrán definidas en la tabla **Relation**, donde tendremos una clave externa para la entidad origen y otra para la entidad destino y un campo tipo para diferenciar si la relación es una asociación, una extensión o una inclusión.

Por ultimo, para hacer un seguimiento de los cambios realizados en todas las tablas relacionadas con los diagramas, se ha creado la entidad **Changes**, que guardará el cambio realizado junto con la fecha y el autor para mostrarlo al usuario.

4 Desarrollo

Aquí se muestra un recopilatorio de las tareas realizadas y los problemas encontrados durante el desarrollo de cada aplicación junto con la resolución que se ha encontrado en cada caso.

4.1 Desarrollo aplicación servidora

Para la API, se ha escogido Java, y en concreto el Framework Spring. En principio, por la familiaridad con el lenguaje, pero también en parte por la disponibilidad de paquetes y complementos.

Se ha instalado la versión 15.0.2 del JDK y como IDE se ha elegido Intellij de IDEA, ya que es un IDE muy completo y robusto, aunque un poco exigente en recursos, en concreto por su consumo de RAM.

Para el motor de base de datos, se utilizará MYSQL, en su versión 8.0.23. Como cliente, en principio se escogió DBeaver, un cliente que además de gratuito, es multiplataforma. Finalmente, al contar con la versión Ultimate de IntelliJ, que tiene gestor de base de datos, fue mas conveniente el uso de este, ya que además de la comodidad de tener ambos usos centralizados en una aplicación, se evitaba el uso concurrente de varios programas, con el consumo de recursos que esto suponía.

Para inicializar el proyecto, se utilizó el gestor de aplicaciones de Spring *Spring Initializer* [28] y se cargaron las dependencias que se utilizarían mas adelante. En concreto:

- Spring Web, para gestionar REST y usar Tomcat como contenedor.
- JPA para persistencia en BBDD.
- MySQL Driver. Para la conexión con el motor escogido.

Se utiliza Maven para la gestión del proyecto y se procede a activar el control de versiones con GIT y almacenaje remoto del código en GitLab, aparte de para llevar un control de los cambios, como medida de seguridad ante perdida del Código:
https://git.eps.uam.es/federico.baras/usecaseapi.git

A parte, con vistas a la realización de las pruebas, se instala también la aplicación Postman y HTTPie, aplicaciones que dan la posibilidad de realizar peticiones HTTP para poner a prueba el servicio.

Una vez preparado el entorno, se procede a codificar, siguiendo un esquema basado en la arquitectura Controlador-Servicio-Repositorio.

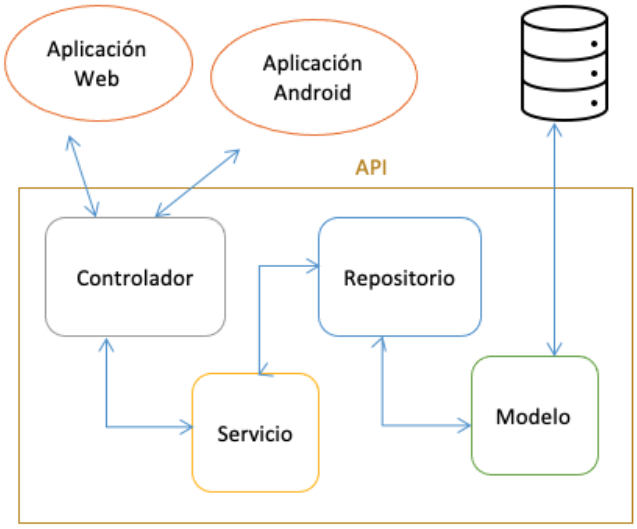

Figura 6: Arquitectura de la aplicación servidora.

Según se aprecia en el esquema, las peticiones de nuestras aplicaciones Web y Android, entran a la API a través del controlador.

- **Controlador**: Mapea la petición del usuario a las funciones especificas pasando la información proporcionada por este a la capa de servicio para aplicar la lógica de negocio.
- **Servicio**: Recibe los datos del controlador y ejecuta la lógica de negocio y la lógica de validación y llama al repositorio.
- **Repositorio**: Es la capa que interactúa con la base de datos a través del modelo.
- **Modelo**: Son las clases que interactúan con la base de datos.

De cara a satisfacer el **RNF 1.2**, se procede a discriminar el acceso a lo métodos de controlador según el rol del usuario. Esto se consigue por medio de la etiqueta @PreAuthorize, que está disponible gracias al paquete Security de Spring.

Otra medida de seguridad, que nos valdrá para cumplir con el **RNF 1.1**, y en la que también participa el paquete Security de Spring, es la gestión de la autenticación por medio de contraseña con JWT [29]. Esto nos permitirá que, una vez autenticado con éxito, se le proporcione la contraseña al usuario, que deberá adjuntar a cada petición al servicio, logrando con ello un mecanismo de autenticación sin estado -*stateless*- reduciendo con ello la necesidad de estar consultando la base de datos múltiples veces. También permite fijar un limite de tiempo para la duración de la sesión.

Una vez creada la estructura básica y a la hora de integrarlo con la aplicación web, surge el problema de que, por seguridad, los navegadores prohíben las llamadas a recursos que se encuentran fuera del origen actual. Para solucionar esto, utilizaremos CORS (Cross-Origin

Resource Sharing), que es un mecanismo que, a través de las cabeceras de los encabezados HTTP, va a permitir a un determinado cliente acceder a los recursos de un servidor diferente al del servidor actual. Esto en Spring se implementa muy fácilmente, bien por medio de anotaciones, o utilizando filtros. En este caso, se utilizará un filtro.

```
@Bean
public CorsFilter corsFilter() {
    CorsConfiguration corsConfiguration = new CorsConfiguration();
    corsConfiguration.setAllowCredentials(true);
    corsConfiguration.setAllowedOrigins(Arrays.asList("http://localhost:4200"));
    corsConfiguration.setAllowedHeaders(Arrays.asList("Origin", "Access-Control-Allow-Origin", "Content-Type",
            "Accept", "Authorization", "Origin, Accept", "X-Requested-With",
            "Access-Control-Request-Method", "Access-Control-Request-Headers"));
    corsConfiguration.setExposedHeaders(Arrays.asList("Origin", "Content-Type", "Accept", "Authorization",
            "Access-Control-Allow-Origin", "Access-Control-Allow-Origin", "Access-Control-Allow-Credentials"));
    corsConfiguration.setAllowedMethods(Arrays.asList("GET", "POST", "PUT", "DELETE", "OPTIONS"));
    UrlBasedCorsConfigurationSource urlBasedCorsConfigurationSource = new UrlBasedCorsConfigurationSource();
    urlBasedCorsConfigurationSource.registerCorsConfiguration( pattern: "/**", corsConfiguration);
    return new CorsFilter(urlBasedCorsConfigurationSource);
}
```

Figura 7: Código CORS

Uno de los problemas principales que puede tener una aplicación colaborativa es la gestión del acceso concurrente a la base datos.

Para conseguir esto, debemos hacer uso del mecanismo de búsqueda optimista proporcionado por *Java Persistence API*.

Utilizaremos la anotación *@Version* en la entidad que queremos proteger, en este caso diagrama. De esta forma, cada transacción que lee datos guarda el valor de la propiedad versión y antes de que la transacción intente una actualización, verifica la versión de la propiedad. Si el valor ha cambiado entre una consulta y otra, se lanza una excepción *OptimisticLockException*.

4.2 Desarrollo aplicación web

Para el desarrollo Web, se ha optado por la utilización de Angular. Para ello, se procede a instalar Angular. Utilizando el gestor de paquetes NPM, se instalará Angular, junto a NodeJS. También se instalará Bootstrap como librería para componentes de la interfaz y FontAwesome, para el estilo de los iconos.

Se utilizará como IDE para este desarrollo el Visual Studio Code, que además de gratuito, implementa fácilmente el control de versiones y tiene un buen depurador.

Se activa el control de versiones y se vincula con el repositorio del proyecto: https://git.eps.uam.es/federico.baras/usecaseweb.git

Con el entorno funcionando, se pasa a codificar la aplicación siguiendo este esquema:

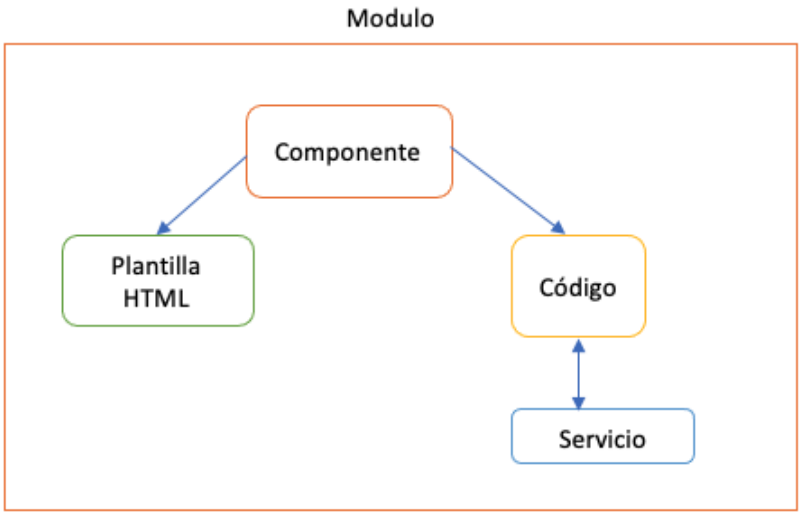

Figura 8: Arquitectura de la aplicación Web.

Los módulos están formados por componentes, el modelo y los servicios.
El componente a su vez está formado por la plantilla HTML por un lado, y el código por otro, siendo en este caso Typescript.

El código, contiene los datos del programa y la lógica y está asociado con la plantilla, formada por HTML y código Angular que enlaza la interfaz con los datos.
Existen dos tipos de enlace:

- Enlace de eventos, en los que la aplicación responde a la interacción del usuario y actualiza los datos de la aplicación.
- Enlace de propiedades, en los que son los cambios en los datos lo que provoca la actualización de los datos que se muestran al usuario.

También se pueden dar los dos tipos de enlace a la vez, ya que Angular soporta el enlace bidireccional.

Por otra parte, tenemos los servicios, que serán los encargados de gestionar las llamadas a la API REST. La declaración de las clases de servicio, vendrán precedidas de un decorador @Injectable que permite inyectarlo como dependencias en otra clase, normalmente en el código del componente correspondiente.

Una vez codificado lo básico, un primer problema que ha sido necesario solucionar es adaptar el código para que las peticiones a la API vayan en consonancia con lo que esta espera, teniendo en cuenta que, por seguridad, esta se ha protegido con autenticación por contraseña.

Para ello, en la primera petición durante el inicio de sesión, en caso de que esta sea satisfactoria, se recogerá la contraseña devuelta por el servicio, que habrá que guardar para adjuntarla en la cabecera de cada una de las siguientes peticiones.

Para simplificar esto, se codificará un *interceptor* que capturará las llamadas a la API añadiendo la cabecera requerida.

```
const TOKEN_HEADER_KEY = 'Authorization';       // para Spring Boot back-end

@Injectable()
export class AuthInterceptor implements HttpInterceptor {
  constructor(private token: TokenStorageService) { }

  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
    let authReq = req;
    const token = this.token.getToken();
    if (token != null) {
      authReq = req.clone({ headers: req.headers.set(TOKEN_HEADER_KEY, 'Bearer ' + token) });
    }
    return next.handle(authReq);
  }
}

export const authInterceptorProviders = [
  { provide: HTTP_INTERCEPTORS, useClass: AuthInterceptor, multi: true }
];
```

Figura 9: Código Interceptor.

Esta clase será también @Injectable, ya que se inyectará en el modulo principal de la aplicación.

Para la gestión de los usuarios, se ha optado por un diseño que muestra los usuarios en tarjetas con botones que dan acceso a los modales de edición y creación.

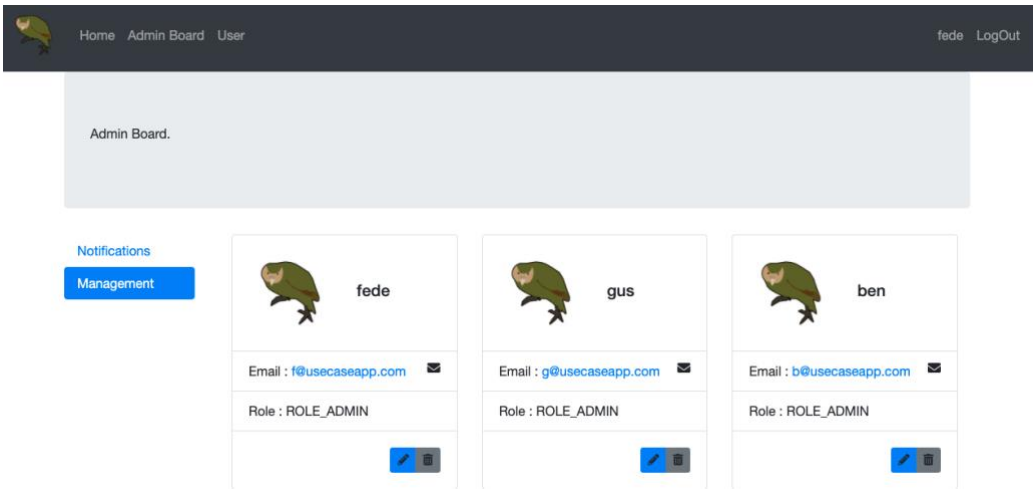

Figura 10: Desarrollo web, usuarios.

Esta captura muestra la página web con el tablero de administrador activo. Tiene un menú vertical de tipo pestañas, que en este caso muestra la edición de usuarios.
Para el diseño, se ha utilizado tanto Bootstrap a modo mas general, como FontAwesome para los iconos.

Para la visualización de los diagramas, se ha hecho uso de la librería Ngx-Graph, con el siguiente resultado:

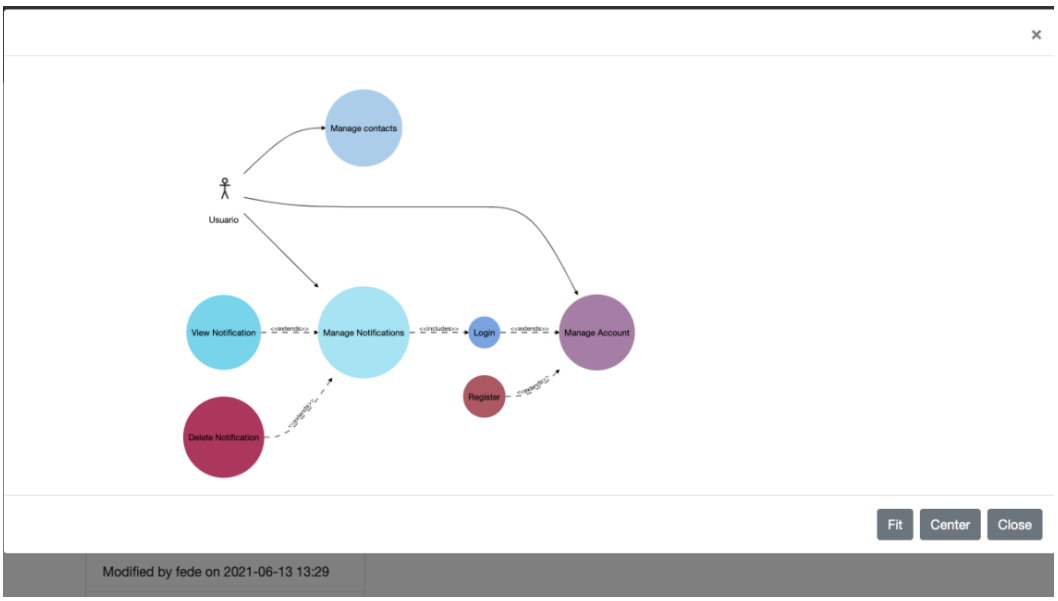

Figura 11: Desarrollo web, vista.

El *layout* del diagrama, permite ampliar o disminuir la imagen y mover los componentes de sitio. También tiene un botón para adaptarlo al contenedor y otro para centrarlo.

4.3 Desarrollo de la aplicación Android

Dado que la aplicación móvil va a ser en Android, la elección lógica del IDE de desarrollo sería en principio Android Studio, ya que, aparte de que está basado en IntelliJ, con el que se ha trabajado satisfactoriamente en el desarrollo de la API, está muy enfocado a la plataforma Android, con lo que hace mas sencillo el trabajo.

Se activa el control de versiones y se vincula con el repositorio del proyecto, accesible mediante la url:
https://git.eps.uam.es/federico.baras/usecaseandroid

Aunque había duda antes de empezar el desarrollo entre Kotlin y Java, finalmente el lenguaje de programación para la aplicación móvil será Java.

Una vez inicializada la aplicación, se instalarán una serie de paquetes que serán convenientes mas adelante:

- Retrofit: Librería para para gestionar las llamadas a la API.
- GSON: Librería para serializar la respuesta JSON.
- Material: Componentes y herramientas para el diseño de la interfaz.

A la hora de codificar, se seguirá generalmente una arquitectura del estilo:

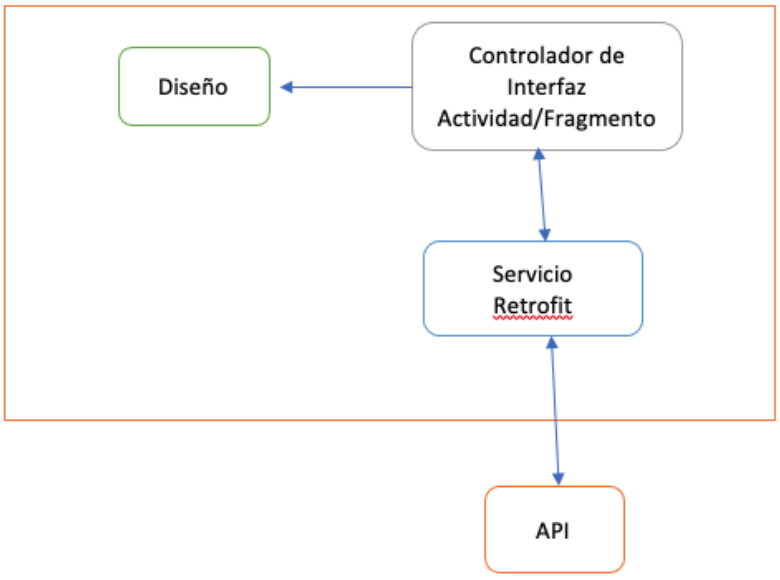

Figura 12: Arquitectura aplicación móvil.

El componente principal es la actividad. Cuando se inicia, se enlaza con el *layout*, que contiene los elementos de la interfaz. Así, cada el elemento del *layout,* como listas, botones

y campos de texto, se enlaza con su elemento correspondiente del controlador. A estos componentes se le emparejan eventos, como por ejemplo el presionar un botón, con una serie de acciones. Muchas veces esto implica hacer una consulta a la API. De esto se encarga la librería de Retrofit, que simplifica el efectuar llamadas de red y obtener una respuesta estructurada.

Ejemplo de llamada con Retrofit:

```
Retrofit retrofit = new Retrofit.Builder()
        .baseUrl("http://192.168.1.99:8080/")
        .addConverterFactory(GsonConverterFactory.create())
        .build();
```

Figura 13: Código Retrofit A.

```
private void getNotifications() {

    //Le paso el token en la llamada. Si veo que lo voy a usar mucho, tendré que meter un Interceptor
    String authToken = "Bearer " + user.getToken();
    Call<List<Notification>> call = notificationService.getNotificationsByReceiver(authToken, user.getId());

    call.enqueue(new Callback<List<Notification>>() {
        @Override
        public void onResponse(Call<List<Notification>> call, Response<List<Notification>> response) {
            if (!listNotifications.isEmpty()) listNotifications.clear();
            notifications = response.body();
            for(Notification post : response.body()) {
                listNotifications.add(StringUtils.substring(post.getNotifText(), start: 0, end: 40));
            }
            arrayAdapter.notifyDataSetChanged();
        }
        @Override
        public void onFailure(Call<List<Notification>> call, Throwable t) {
            Toast.makeText(getActivity(),  text: "Error getting notifications.", Toast.LENGTH_SHORT).show();
        }
    });
}
```

Figura 14: Código Retrofit B.

Trabajando con Android Studio, nos encontramos que tenemos bastantes recursos disponibles. Así, encontramos que tenemos algunas plantillas que podemos utilizar con pequeñas adaptaciones, como por ejemplo la actividad del inicio de sesión y la actividad del menú lateral.

En esta parte del desarrollo, la parte mas complicada ha sido la elección de un método para la creación de diagramas que mantuviese un equilibrio entre lo intuitivo de el modelado gestual y un modelo ligero que permitiese el trabajo cooperativo y multiplataforma.

Al contar con la posibilidad de mostrar el diagrama en la aplicación web, para la aplicación móvil se ha optado por un diseño que puede que no sea tan intuitivo como una interfaz de diseño gestual, pero que es ligero y consistente.

La pagina principal del apartado de diagramas contiene una lista con tarjetas que simbolizan cada diagrama, con el nombre, fecha de creación y modificación y el usuario involucrado. En la barra inferior, se sitúan los botones para agregar o eliminar diagramas.

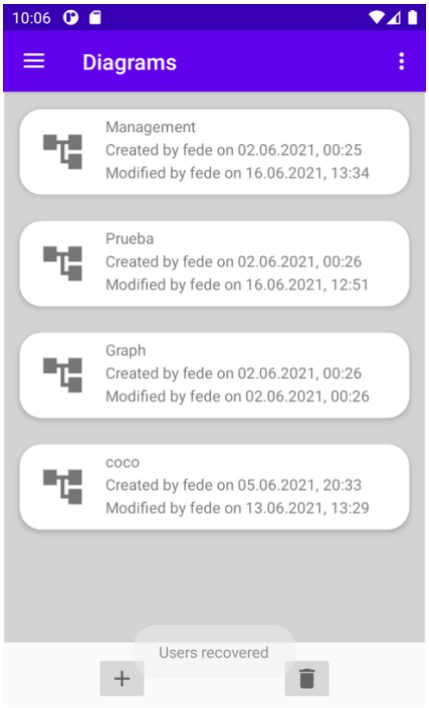

Figura 15: Desarrollo móvil. Lista de diagramas.

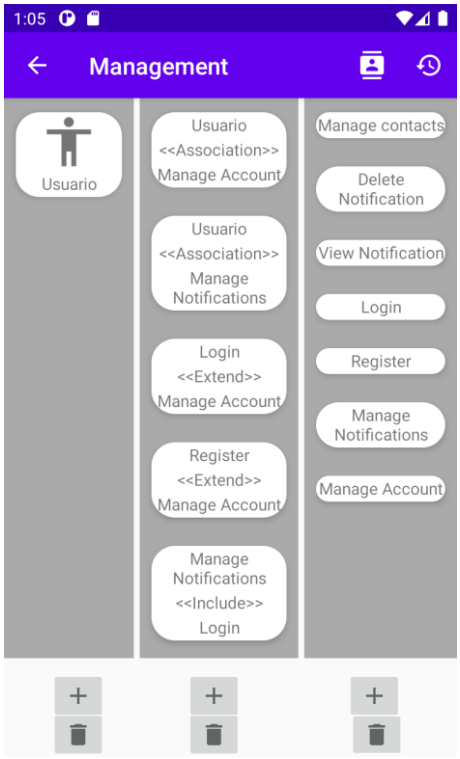

Figura 16: Desarrollo móvil. Detalle de diagrama.

Se ha intentado condensar la gestión del modelado del diagrama en una sola actividad. Se ha dividido la pantalla en tres columnas. La primera muestra la entidad *Actor*, con los botones para añadir y eliminar. La segunda columna es la relativa a las relaciones y la tercera esta destinada a las entidades de tipo "Caso de Uso". Los botones para agregar o eliminar casos de uso, al igual que los relativos al actor, gestionan las acciones en la misma actividad a través de ventanas emergentes, mientras que los botones para agregar o eliminar relaciones, nos redirigen a una nueva actividad, que nos mostrará las distintas listas con los elementos. En la barra de herramientas superior, tendremos acceso a los botones para gestionar los participantes del diagrama y para ver la lista de cambios.

Todo este funcionamiento está mas extensamente explicado en el anexo del manual.

5 Integración y pruebas

En este capitulo se detallará el proceso de integración entre las tres aplicaciones, las pruebas realizadas y el alcance y resultado de estas.

5.1 Integración

Debido a que la gestión de datos está centralizada en el servidor, la integración se ha centrado en gran medida en la interacción de las aplicaciones móvil y web con la aplicación servidora.

Aunque durante la fase de análisis se definió el modelo de datos, durante las sucesivas iteraciones de la implementación, se introdujeron pequeñas modificaciones que hicieron necesario hacer variaciones de este. Como consecuencia, hubo que hacer cambios en la aplicación servidora y propagar estos cambios a las aplicaciones web y móvil, que consumen los servicios que proporciona esta.

Para la integración del servidor con la base de datos se ha utilizado JDBC y *Spring Data JPA*, que nos permite hacer muchas de las gestiones con simples anotaciones. De hecho, no ha sido necesario escribir a mano ninguna de las consultas a la base de datos. También esta automatizado de esta manera la creación de tablas y campos.

5.2 Pruebas unitarias

En este apartado se mostrarán las pruebas unitarias realizadas a cada bloque. En este caso, se trata de pruebas de los casos de uso derivados del análisis de requisitos. Cada caso de uso está relacionado con un formulario. Las pruebas se centran en introducir datos que podrían dar lugar a error en los campos del formulario, enviar el formulario y analizar la salida para comprobar si los resultados son acordes con el resultado esperado.

La primera columna “Rol”, indica el tipo de usuario que puede realizar la acción. La columna “Formulario”, se refiere al formulario relacionado con el caso de uso correspondiente. En la columna “Campos” se listan los campos del formulario. La columna “Pruebas” detalla las pruebas a las que se someterá a los campos del formulario. En la columna “Salida”, se muestra el resultado esperado por cada prueba y por ultimo, la columna “Resultado” tiene como objetivo informar si se ha superado la prueba.

- Gestión de cuenta.

Rol	Formulario	Campos	Pruebas	Salida	Resultado
User / Admin	Registro	- Nombre usuario - Correo -Contraseña	-Introducir usuario existente -Introducir email existente -Email con formato incorrecto -Dejar algún campo vacío -Introducir contraseña de menos de 6 caracteres	-Error: El usuario ya existe -Error: El email ya existe -Error: El formato del email es incorrecto -Error: los campos son obligatorios -Error: La contraseña debe tener al menos 6 caracteres	OK
	Login	-Nombre de usuario -Contraseña	-Dejar algún campo vacío -Usuario y contraseña no coinciden -Introducir contraseña de menos de 6 caracteres	-Error: los campos son obligatorios -Error: El inicio de sesión falló -Error: La contraseña debe tener al menos 6 caracteres	OK

Tabla 1: Pruebas: gestión de usuarios

- Gestión de notificaciones

Rol	Formulario	Campos	Pruebas	Salida	Resultado
User / Admin	Ver notificación	-Lista notificaciones	-Agregar notificación sin seleccionar elemento	-Botón desactivado	OK
	Eliminar notificación	-Lista notificaciones	-Agregar notificación sin seleccionar elemento	-Botón desactivado	OK

Tabla 2: Pruebas: gestión de notificaciones.

- Gestión de contactos

Rol	Formulario	Campos	Pruebas	Salida	Resultado
User / Admin	Agregar contacto	-Email contacto	-Agregar email vacío -Agregar email de usuario inexistente en el sistema -Agregar email de usuario que ya es contacto	-Error: Email no puede estar vacío -Error: El email no existe en el sistema -Error: Ese email ya pertenece a tu lista de contactos	OK
	Ver contacto	-Lista contactos	-Agregar contacto sin seleccionar elemento	-Botón desactivado	OK

	Eliminar contacto	-Lista contactos	-Eliminar contacto sin seleccionar elemento	-Botón desactivado	OK
	Aceptar solicitud de amistad	-Lista de solicitudes de amistad	-Aceptar solicitud sin seleccionar elemento	-Botón desactivado	OK

Tabla 3: Pruebas: gestión de contactos

- Gestión de diagramas

Rol	Formulario	Campos	Pruebas	Salida	Resultado
User / Admin	Agregar diagrama	-Nombre diagrama	-Agregar nombre de diagrama vacío -Agregar nombre de diagrama existente	-Error: el nombre no puede estar vacío -Error: el nombre de diagrama no puede estar repetido	OK
	Eliminar diagrama	-Lista diagramas	-Agregar diagrama sin seleccionar elemento	-La lista tiene un elemento seleccionado por defecto	OK

Tabla 4: Pruebas: gestión de diagramas.

- Gestión del detalle del diagrama

Rol	Formulario	Campos	Pruebas	Salida	Resultado
User / Admin	Agregar actor	-Nombre actor	-Agregar nombre de actor vacío -Agregar nombre de actor existente	-Error: el nombre no puede estar vacío -Error: el nombre de actor no puede estar repetido	OK
	Eliminar actor	-Lista actores	-Agregar diagrama sin seleccionar elemento	-La lista tiene un elemento seleccionado por defecto	OK
	Agregar relación	-Lista entidad origen -Lista entidad destino -Lista tipo de relación	-Agregar relación sin seleccionar elemento de alguna de las listas	-El botón se desactiva a menos que las tres listas tengan un elemento seleccionado	OK
	Eliminar relación	-Lista relaciones	-Agregar relación sin seleccionar elemento	- El botón se desactiva a menos haya un elemento seleccionado	OK
	Agregar uso	-Nombre uso	-Agregar nombre de uso vacío -Agregar nombre de uso existente	-Error: el nombre no puede estar vacío -Error: el nombre de uso no puede estar repetido	OK

	Eliminar uso	-Lista usos	-Agregar diagrama sin seleccionar elemento	-La lista tiene un elemento seleccionado por defecto	OK
	Agregar participante	-Lista contactos	-Agregar contacto a la lista de participantes sin seleccionar un elemento	- El botón se desactiva a menos haya un elemento seleccionado	

Tabla 5: Pruebas: gestión del detalle del diagrama.

- Envío de notificaciones

Rol	Formulario	Campos	Pruebas	Salida	Resultado
Admin	Envío notificación	-Texto notificación -Selector usuario	-Texto de notificación vacío -Ningún usuario seleccionado	-Error: el texto no puede estar vacío -Error: seleccione un usuario	OK

Tabla 6: Pruebas: envío de notificaciones.

- Gestión de usuarios

Rol	Formulario	Campos	Pruebas	Salida	Resultado
Admin	Agregar usuario	-Nombre de usuario -Email -Contraseña -Rol	-Escribir nombre de usuario existente - Escribir email existente -Escribir rol en formato incorrecto -Escribir contraseña de menos de 6 caracteres -Dejar alguno de los campos vacíos	-Error: el usuario ya existe -Error: el email ya existe -Error: el rol no tiene el formato correcto Error: la contraseña tiene menos de 6 caracteres -Error: campo obligatorio	OK
	Editar usuario	-Nombre de usuario -Email-Rol	-Agregar nombre de usuario existente -Agregar email existente -Escribir rol en formato incorrecto -Dejar alguno de los campos vacíos	-Error: el usuario ya existe -Error: el email ya existe -Error: el rol no tiene el formato correcto -Error: campo obligatorio	OK

Tabla 7: Pruebas: gestión de usuarios.

5.3 Pruebas de rendimiento y carga

La aplicación se ha probado con hasta 5 usuarios colaborando simultáneamente en un diagrama.

Las pruebas han consistido en operaciones simultaneas sobre el mismo diagrama para comprobar la corrección de la persistencia.

Así mismo, se comprueba que las operaciones con la base de datos no tardan nunca mas de dos segundos en ser realizadas.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Al final del desarrollo se ha conseguido tener una aplicación servidora segura, con control de acceso por contraseña, sesión y distintos roles para el acceso a los métodos, una aplicación móvil que permite el modelado colaborativo y con funciones adicionales como notificaciones e historial de cambios y una aplicación web, que permite gestionar los usuarios y las notificaciones y permite visualizar los diagramas. Las aplicaciones interactúan fluidamente entre ellas y cuentan con gran modularidad, lo que permite ampliar el desarrollo en un futuro.

Aunque ni la tecnología ni los requerimientos eran excesivamente complejos, el trabajo ha sido bastante extenso, ya que ha sido necesario crear tres aplicaciones para tres plataformas distintas integradas entre si.

6.2 Trabajo futuro

Durante el desarrollo, han ido surgiendo ideas para mejorar la aplicación que por tiempo no se han podido implementar.

Una de ellas, podría ser utilizar el listado de cambios de los diagramas para sacar estadísticas de uso de la aplicación por usuario, generando gráficas para mostrar estos aspectos detalladamente y de un modo mas intuitivo que una lista.

Aunque para el modelado de los diagramas se ha evitado utilizar librerías de gráficos para no restar velocidad de reacción a la aplicación, el resultado no es el optimo, sobre todo en cuanto al aspecto visual, por lo cual, como tarea a futuro, se podría introducir un método para realizar el modelado visualmente.

Referencias

[1] Magin, M., & Kopf, S. (2013). A collaborative multi-touch uml design tool. *Technical reports*, *13*.

[2] Lemma, R., Lanza, M., & Olivero, F. (2013, May). CEL: modeling everywhere. In *2013 35th International Conference on Software Engineering (ICSE)* (pp. 1323-1326). IEEE.

[3] Wüest, D., Seyff, N., & Glinz, M. (2012, October). Flexisketch: A mobile sketching tool for software modeling. In *International Conference on Mobile Computing, Applications, and Services* (pp. 225-244). Springer, Berlin, Heidelberg.

[4] Cacoo, https://cacoo.com/ Última fecha de acceso:05/06/2021]

[5] UMLetino, https://www.umletino.com/ Última fecha de acceso:05/06/2021]

[6] UMLet, https://www.umlet.com/ Última fecha de acceso:05/06/2021]

[7] Visual Paradigm, https://www.visual-paradigm.com/ Última fecha de acceso:05/06/2021]

[8] Lucidchart, https://www.lucidchart.com/pages/ Última fecha de acceso:05/06/2021]

[9] GitLab, https://about.gitlab.com/ [Última fecha de acceso:05/06/2021]

[10] Apache Tomcat, http://tomcat.apache.org/ [Última fecha de acceso:05/06/2021]

[11] JSON, https://www.json.org/json-es.html [Última fecha de acceso:05/06/2021]

[12] Java, https://www.java.com/es/ [Última fecha de acceso:05/06/2021]

[13] Spring, https://spring.io/ [Última fecha de acceso:05/06/2021]

[14] MySQL, https://www.mysql.com/ [Última fecha de acceso:05/06/2021]

[15] JWT, https://jwt.io/ [Última fecha de acceso:05/06/2021]

[16] Lombok, https://projectlombok.org/ [Última fecha de acceso:05/06/2021]

[17] IntelliJ IDEA, https://www.jetbrains.com/es-es/idea/ [Última fecha de acceso:05/06/2021]

[18] Maven, https://maven.apache.org/ [Última fecha de acceso:05/06/2021]

[19] DBeaver, https://dbeaver.io/ [Última fecha de acceso:05/06/2021]

[20] Postman, https://www.postman.com/ [Última fecha de acceso:05/06/2021]

[21] HTTPie, https://httpie.io/ [Última fecha de acceso:05/06/2021]

[22] Angular, https://angular.io/ [Última fecha de acceso:05/06/2021]

[23] Bootstrap, https://getbootstrap.com/ [Última fecha de acceso:05/06/2021]

[24] FontAwesome, https://fontawesome.com/ [Última fecha de acceso:05/06/2021]

[25] Ngx-Graph, https://swimlane.github.io/ngx-graph/ [Última fecha de acceso:05/06/2021]

[26] Retrofit, https://square.github.io/retrofit/ [Última fecha de acceso:05/06/2021]

[27] GraphView, https://mvnrepository.com/artifact/de.blox/graphview/0.7.0 [Última fecha de acceso:05/06/2021]

[28] Spring Initializer, https://start.spring.io/ [Última fecha de acceso:05/06/2021]

[29] JWT, https://www.baeldung.com/java-json-web-tokens-jjwt [Última fecha de acceso:05/06/2021]
[30] CORS on Spring, https://spring.io/blog/2015/06/08/cors-support-in-spring-framework [Última fecha de acceso:05/06/2021]

Glosario

API	Application Programming Interface
JSON	JavaScript Object Notation
JWT	Java Web Tokens
IDE	Integrated Development Environment
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
CORS	Cross-origin resource sharing
HTTP	Hypertext Transfer Protocol
JDBC	Java Database Connectivity
JPA	Java Persistence API

Anexos

A Manual de usuario

Al iniciar la aplicación nos encontramos con la pantalla de acceso donde introduciremos usuario y contraseña. El sistema comprueba si el usuario y la contraseña coincide, entrando en el sistema si es así, o mostrando un mensaje de error en caso contrario.

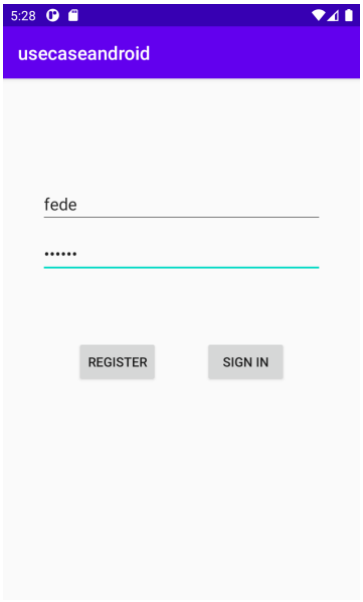

Figura 17: Manual: Login

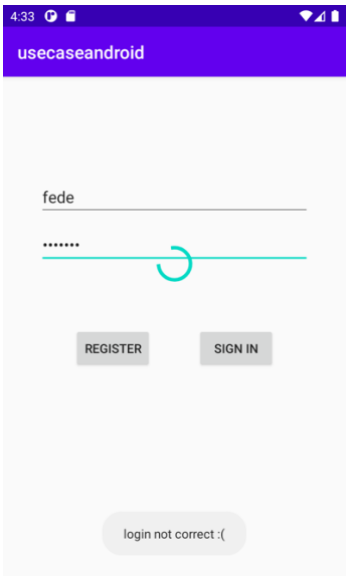

Figura 18: Manual: login error.

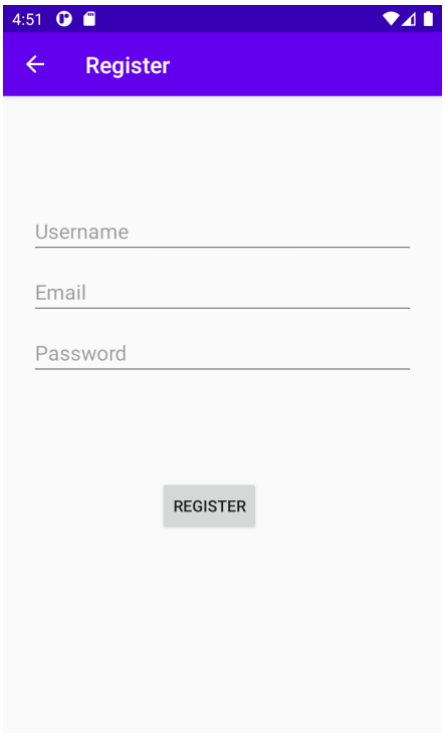

Figura 19: Manual. Registro

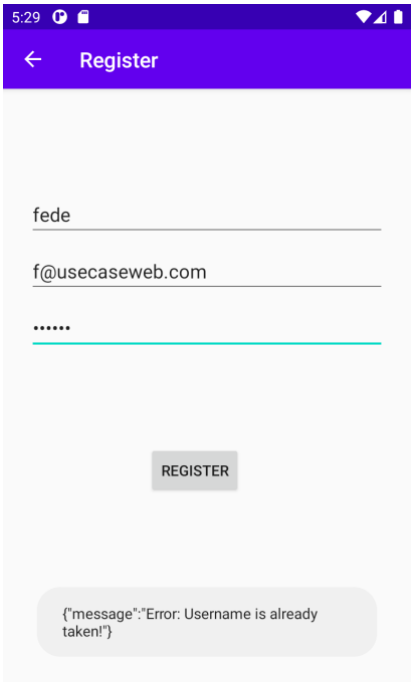

Figura 20: Manual. Registro Error.

De no estar dados de alta en el sistema todavía, habrá que presionar el botón para registrarse y rellenar el formulario. Si alguna de los campos no tiene el formato correcto o el usuario o correo ya existe, se mostrará con un error en pantalla.

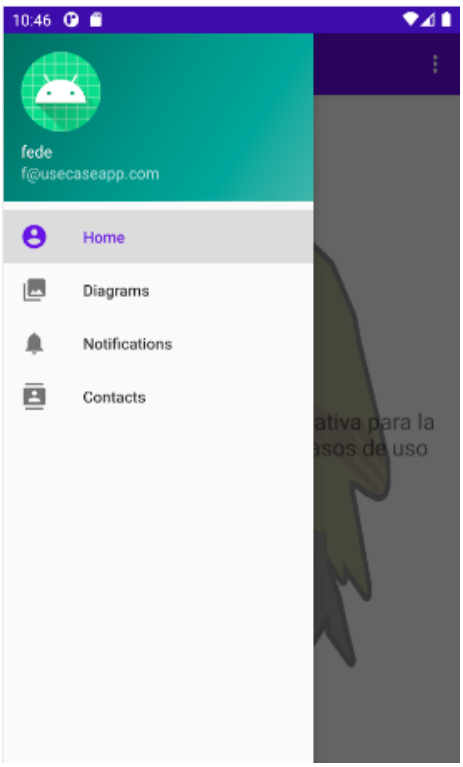

Figura 21 Manual: Inicio fondo.

Figura 22 Manual: Inicio menú.

Una vez autentificado en el sistema, accedemos a la pantalla principal de la aplicación, donde aparece el logo. En la barra de herramientas superior, tenemos acceso al menú lateral, con los accesos a las diferentes secciones de la aplicación.

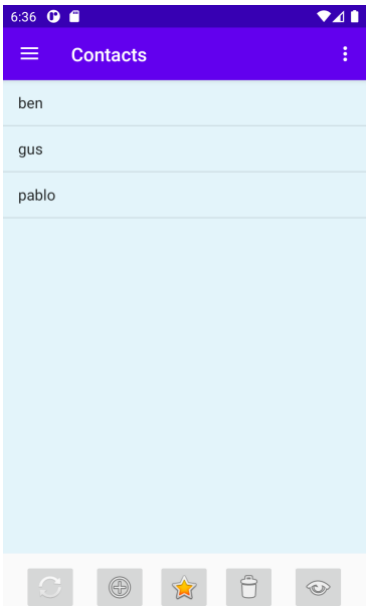

Figura 23: Manual: Contactos.

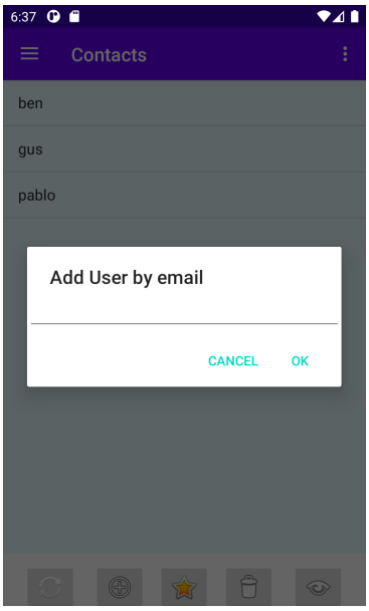

Figura 24: Manual: Contactos añadir.

En la sección de los contactos, tenemos una lista con los contactos asociados al usuario y una serie de botones para actualizar, añadir, eliminar y visualizar el usuario, además de otro botón con forma de estrella que mostrará las solicitudes de amistad pendientes de aceptar.

El botón de agregar usuario lanzará una ventana emergente en la que podremos introducir el email del usuario que queremos agregar. Si el usuario existe, la solicitud quedará pendiente de ser aprobada por el otro usuario.

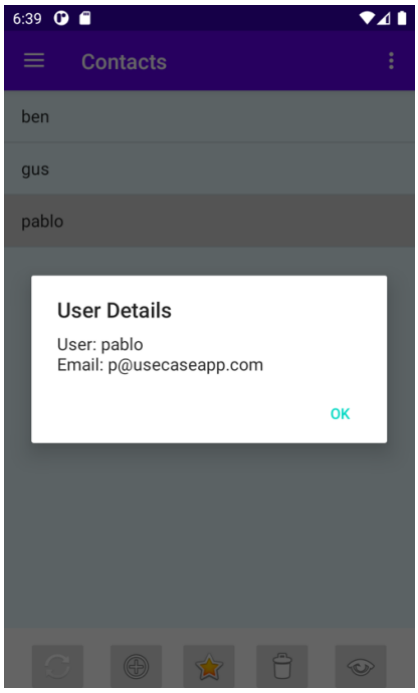

Figura 25: Manual: Contactos mostrar.

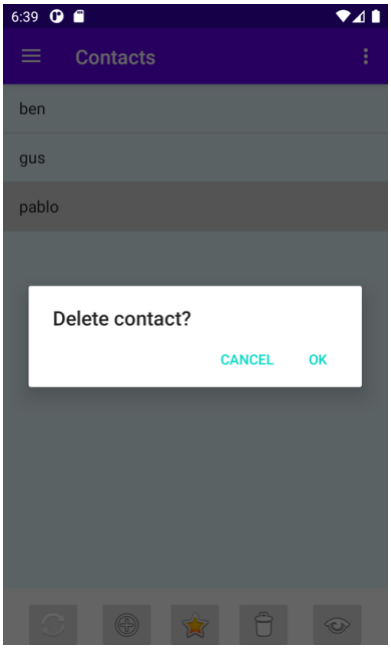

Figura 26: Manual: Contactos eliminar.

Los botones para mostrar o eliminar contactos solamente estarán activos en caso de que haya un usuario seleccionado en la lista de contactos. Al presionar el botón de mostrar, se abrirá una ventana emergente con los detalles del contacto. En el caso del botón eliminar, se nos mostrará una ventana emergente con un dialogo para confirmar la eliminación.

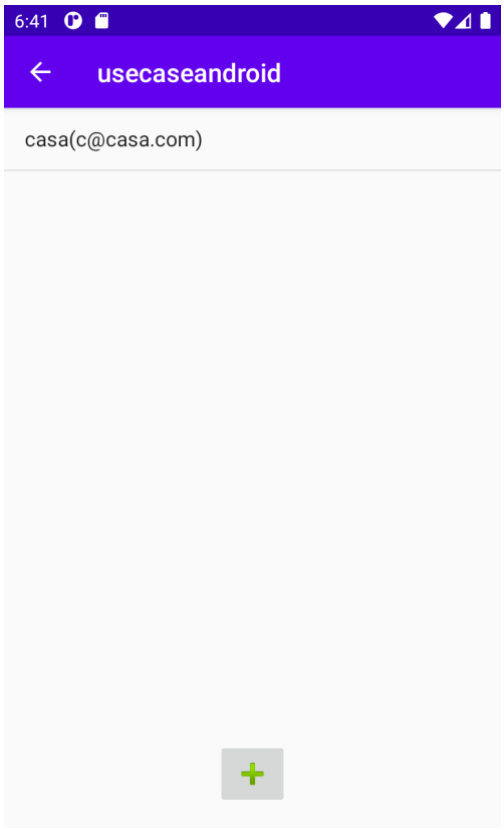

Figura 27: Manual: Contactos solicitud.

El botón para aceptar solicitudes dará paso a otra ventana con una lista con el total de solicitudes pendientes y un botón para confirmar. Al igual que en el caso de los botones para ver y eliminar contactos, el botón para aceptar solicitudes solo estará activo en el caso de que haya algún usuario en la lista y en ese caso, de que alguno se encuentre seleccionado.

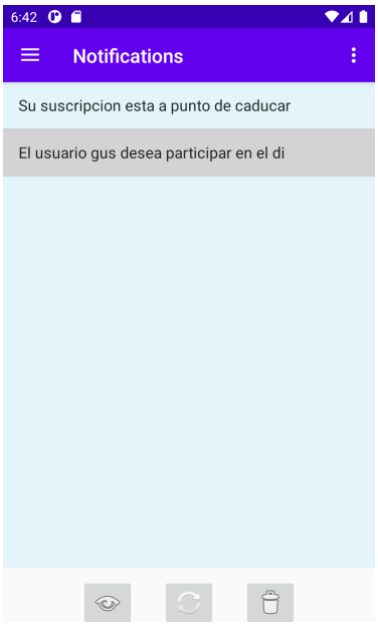

Figura 28: Manual: Notificaciones.

En el apartado de notificaciones, tendremos nuevamente una lista con el conjunto de las notificaciones y los botones para visualizar, actualizar y eliminar.

Como anteriormente, estos no estarán activos hasta que no se seleccione un elemento de la lista.

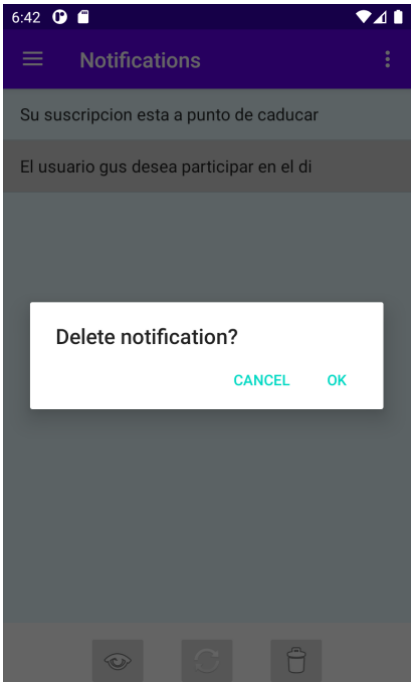

Figura 29: Manual: Notificaciones eliminar.

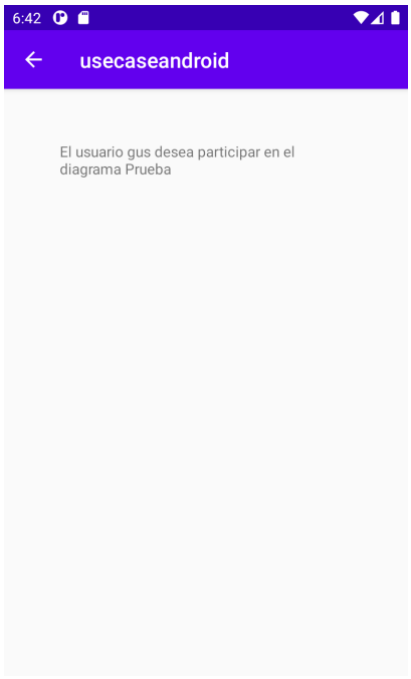

Figura 30: Manual: Notificaciones detalle.

Eliminar lanzará una ventana emergente de confirmación y visualizar llevará a otra actividad donde se mostrará el detalle de la notificación. En esta actividad contaremos con un botón en la barra de herramientas para volver a la actividad anterior.

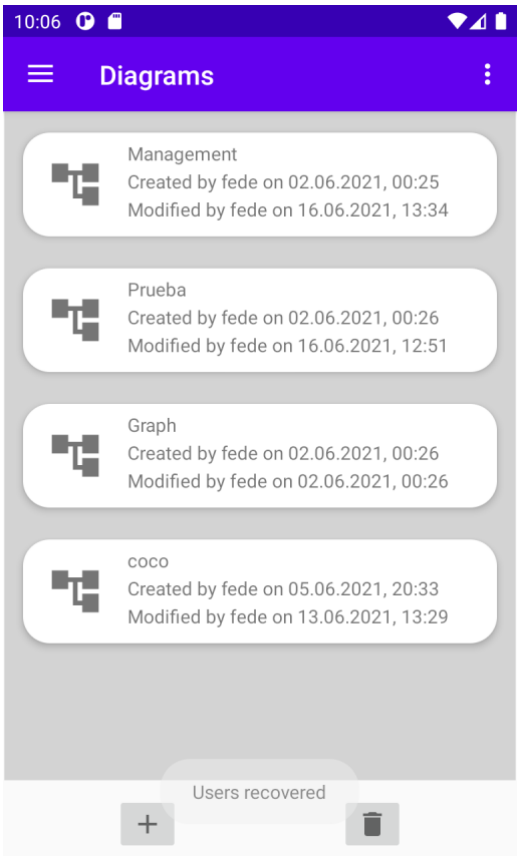

Figura 31: Manual: Diagramas.

El apartado de diagramas mostrará una lista con tarjetas que describen diagramas. En el detalle de estas, a parte de un icono que representa al diagrama, se mostrará el nombre de este y la fecha y el autor de creación junto con la fecha y autor de la ultima modificación.

Tendremos dos botones, para añadir y eliminar diagramas. En ambos casos, al presionarlos, se lanzará una ventana emergente para o bien proporcionar el nombre del nuevo diagrama, o bien seleccionar de una lista el diagrama a eliminar.

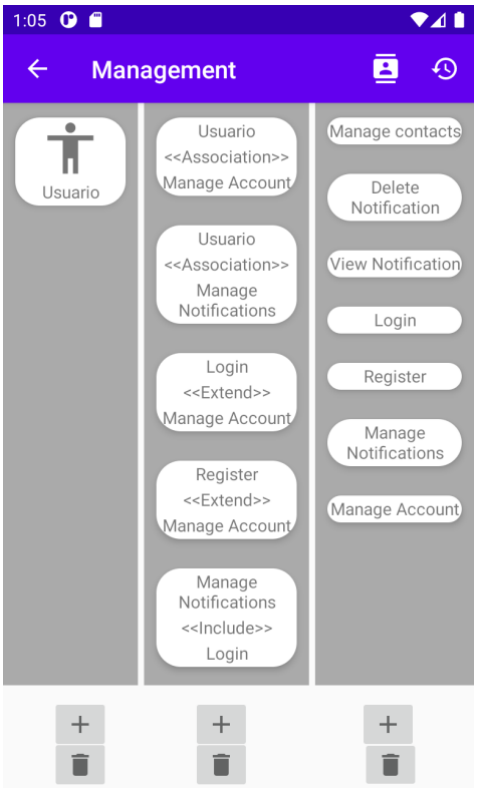

Figura 32: Manual: Diagramas detalle.

La pantalla de detalle del diagrama contará con tres listas, una para actores, otra para relaciones y otra para casos de uso. Las relaciones serán de tipo asociación, inclusión o extensión.

Bajo la columna de actores, al igual que con los casos de uso, tenemos los botones para agregar o eliminarlos. Estos botones lanzan una ventana emergente para introducir el nombre del nuevo actor o bien seleccionar el actor a eliminar.

Los botones para agregar o eliminar relación, darán acceso a una nueva actividad.

En la barra de tareas tendremos los botones para gestionar los participantes y para mostrar el historial de cambios.

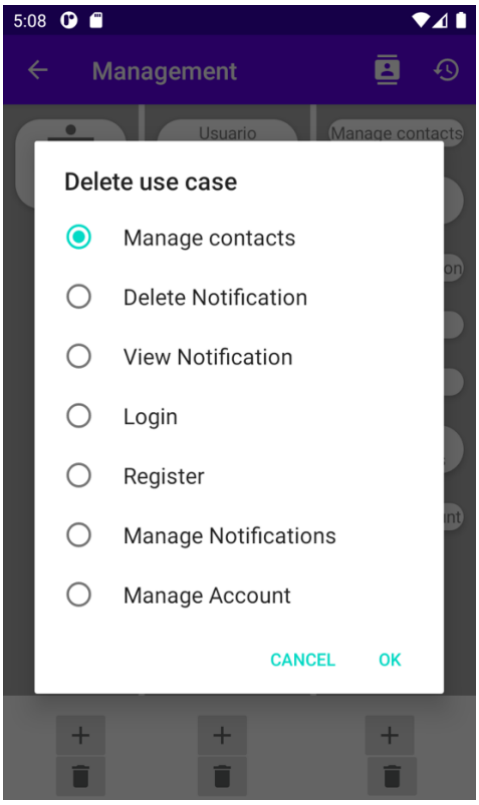

Figura 33: Manual: Diagramas eliminar caso de uso.

Al eliminar o añadir un caso de uso o un actor, se cerrará el modal, y se actualizarán las listas.

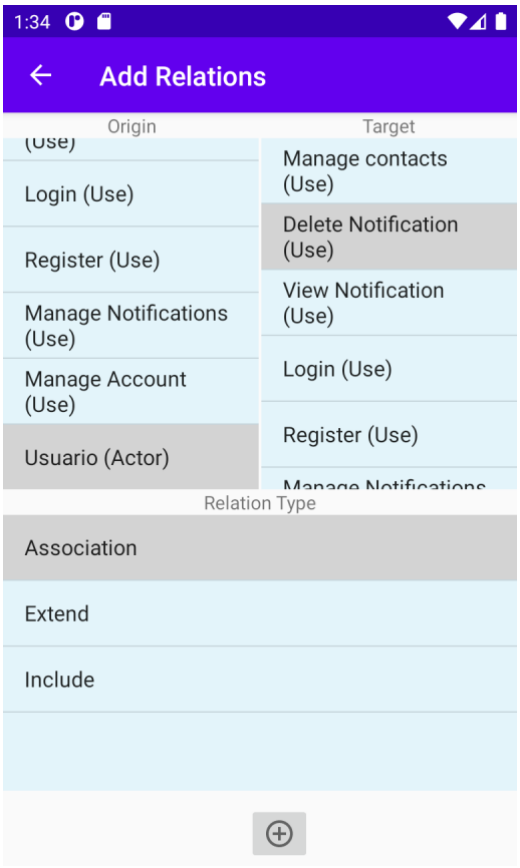

Figura 34: Manual: Diagramas añadir relación.

En la actividad para añadir relaciones, tenemos en la parte superior dos listas, una para la entidad origen y otra para la entidad destino. En la lista inferior, tenemos los tipos de relación que podemos crear entre estas dos entidades. Para que se active el botón, todas las listas deberán tener un elemento seleccionado.

A la hora de añadir una relación se hará una serie de comprobaciones que, de no cumplirse, llevarán a que la aplicación muestre un mensaje de error reseñando el problema. Por ejemplo, una entidad no podrá relacionarse consigo mismo, un actor no puede establecer relación con otro actor, un actor solo puede relacionarse con un caso de uso por medio de una relación de tipo asociación, etc.

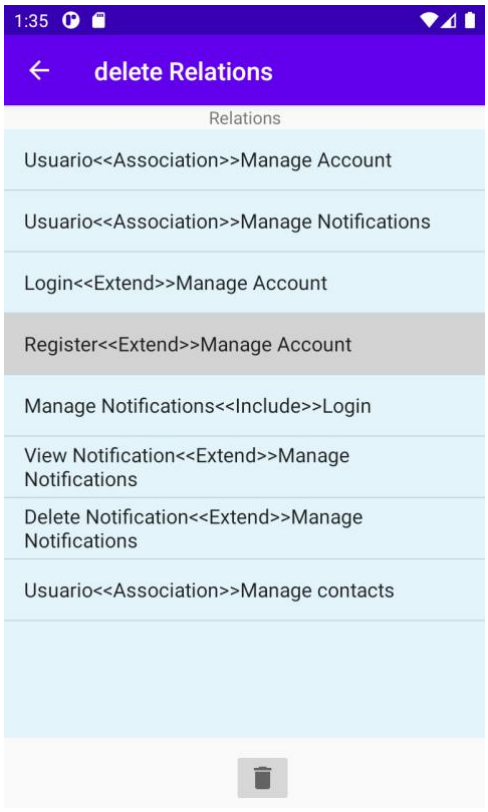

Figura 35: Manual: Diagramas eliminar relación.

La actividad para eliminar relaciones consta de una lista con el conjunto de relaciones. Tendremos que seleccionar una para activar el botón de eliminar. Una vez eliminada, volveremos a la actividad anterior y se habrá actualizado la lista de relaciones.

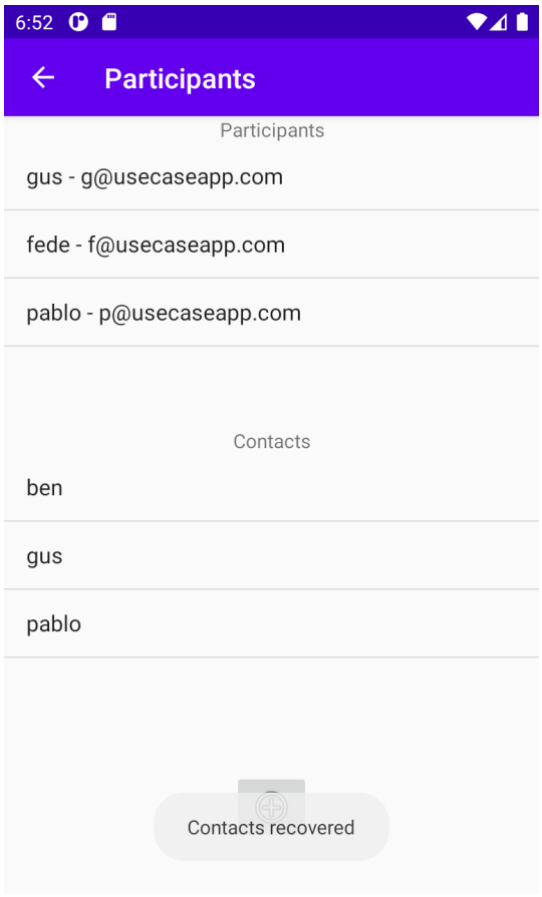

Figura 36: Manual: Diagrama participantes.

La ventana de gestión de participantes mostrará dos tablas. La superior lista los participantes que tienen acceso al diagrama y la inferior muestra la lista de contactos del usuario.

El botón permitirá, si hay algún contacto seleccionado, añadir el contacto a la lista de participantes.

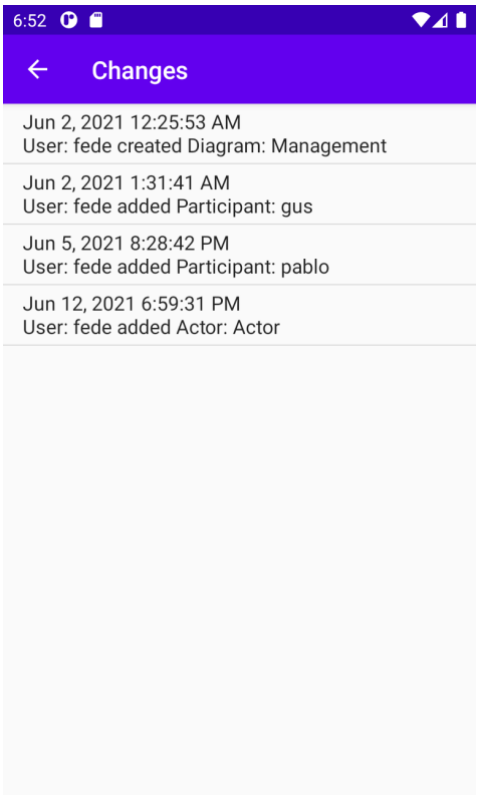

Figura 37: Manual: Diagrama cambios.

La ventana de cambios mostrará un listado de los cambios que se han ido ejecutando sobre el diagrama consecuencia de añadir o eliminar elementos en el mismo.

Mostrará la fecha, el usuario responsable del cambio, la acción ejecutada y el elemento sobre el que se ejecuto dicha acción.

En la interfaz web, tendremos acceso al panel de usuario si entramos como usuario y al panel de administrador y usuario si entramos como administrador.

Figura 38: Manual: Web Inicio Logout.

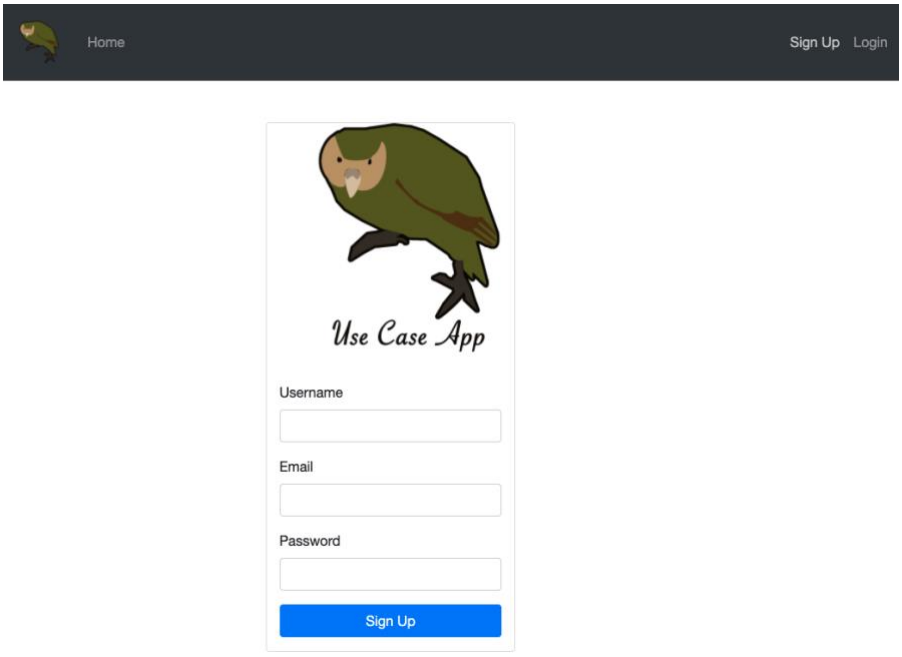

Figura 39: Manual: Web Registro.

La ventana de registro muestra el mismo formulario que en el caso de la aplicación móvil. Igualmente, comprobará que el usuario no exista y que los campos tengan el formato correcto.

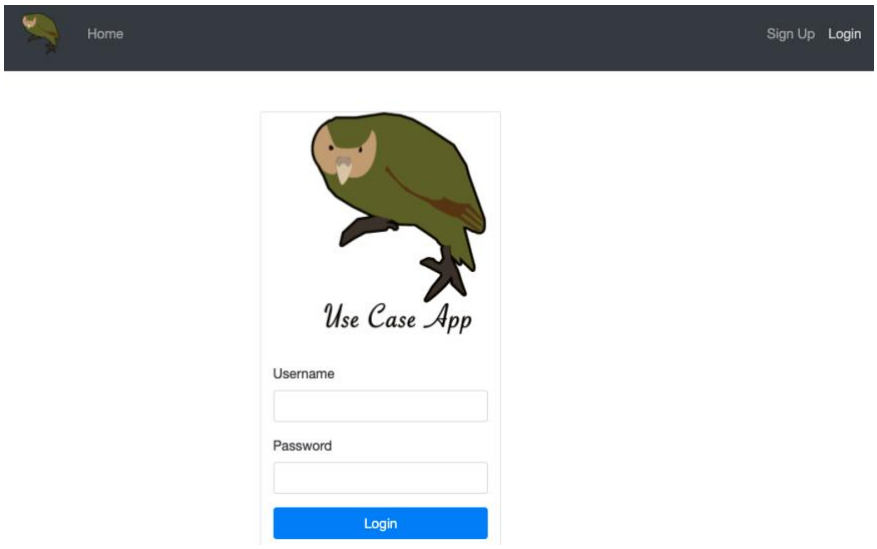

Figura 40: Manual: Web Login.

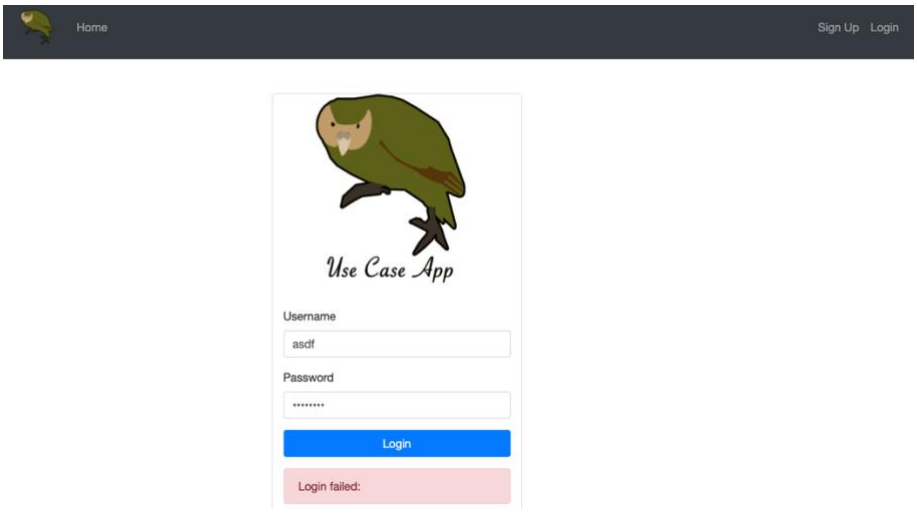

Figura 41: Manual: Web Login error.

La pantalla de inicio de sesión cuenta con un formulario con los campos usuario y contraseña. Comprueba que estos coinciden con los almacenados en el sistema y si no es así, muestra un error.

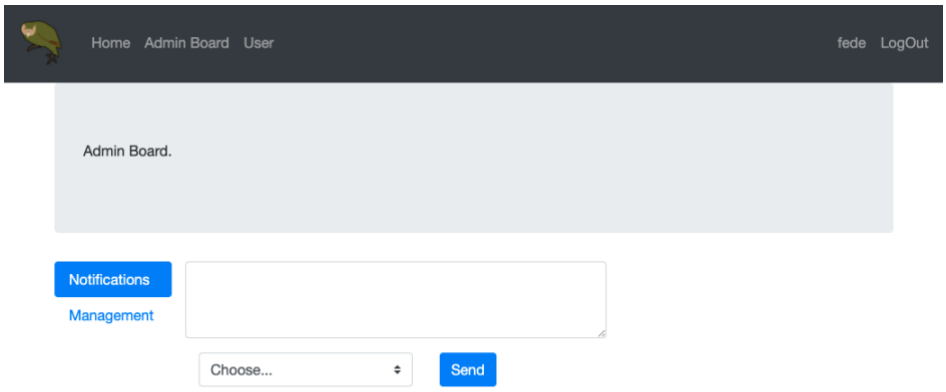

Figura 42: Manual: Web Administrador notificaciones.

Entrando como administrador, tenemos acceso al panel de envío de notificaciones, donde podremos enviar notificaciones a cualquier usuario.

El formulario cuenta con un campo de texto y una lista desplegable para seleccionar el usuario. En caso de que el campo esté vacío o no haya ningún usuario seleccionado, el botón estará inactivo.

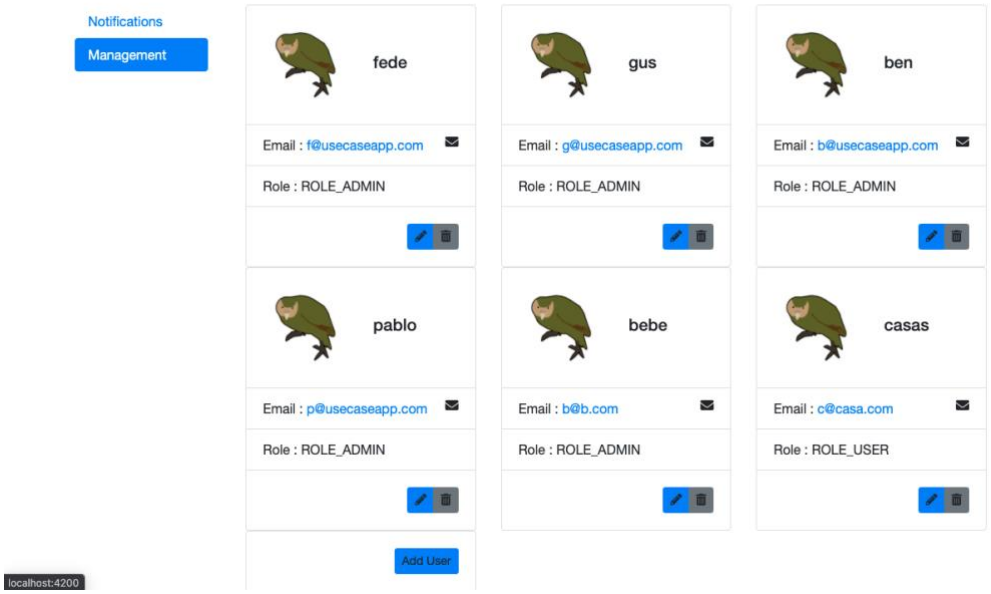

Figura 43: Manual: Web Administrador usuarios.

También como administrador, tenemos acceso al panel de gestión de usuarios.

Se nos mostrará una lista de tarjetas con los datos de los usuarios y los botones de edición y eliminación en cada una de ellas.

La ultima tarjeta de la lista incluye el botón para agregar un nuevo usuario.

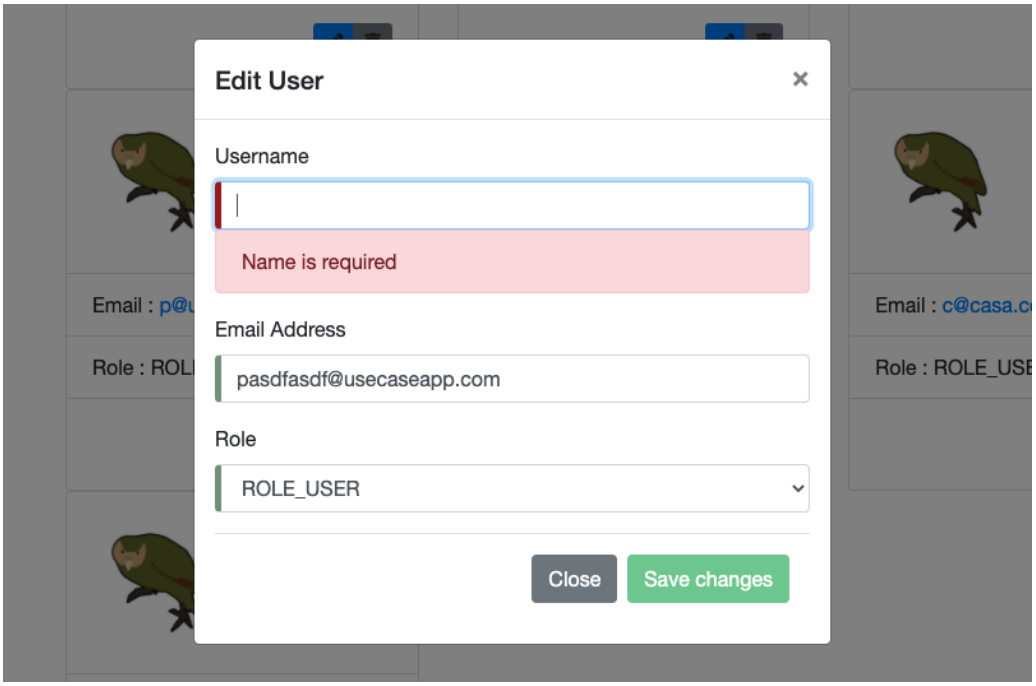

Figura 44: Manual: Web Administrador usuarios editar.

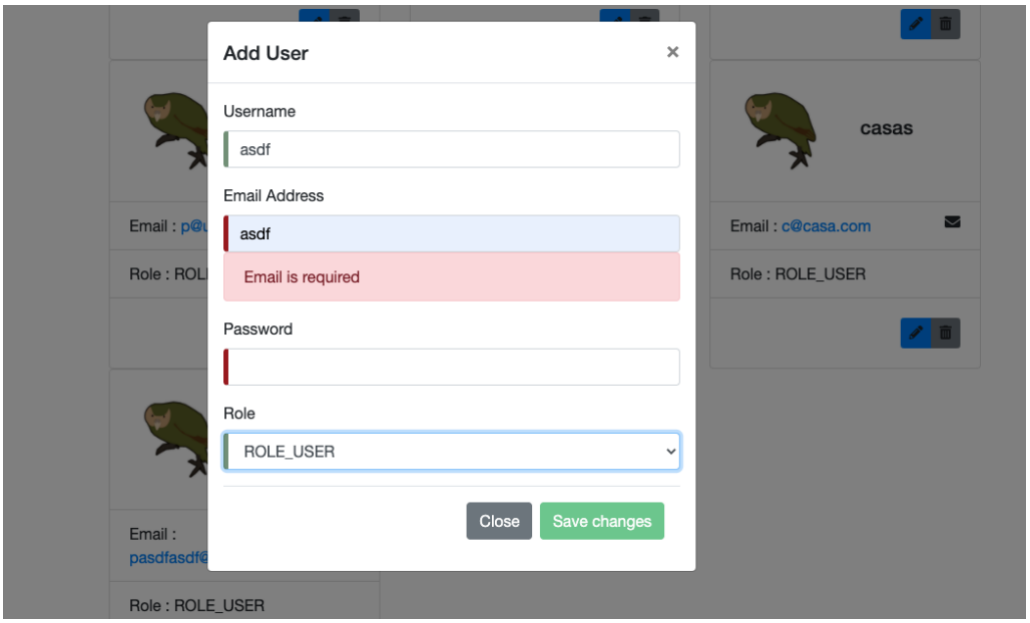

Figura 45: Manual: Web Administrador usuarios agregar.

En el caso de añadir o editar usuario, se nos abrirá un modal similar para ambos casos, que habrá que rellenar y presionar el botón para guardar.

En caso de usuario repetido o errores en el formato de los campos, se nos mostrará un mensaje de error.

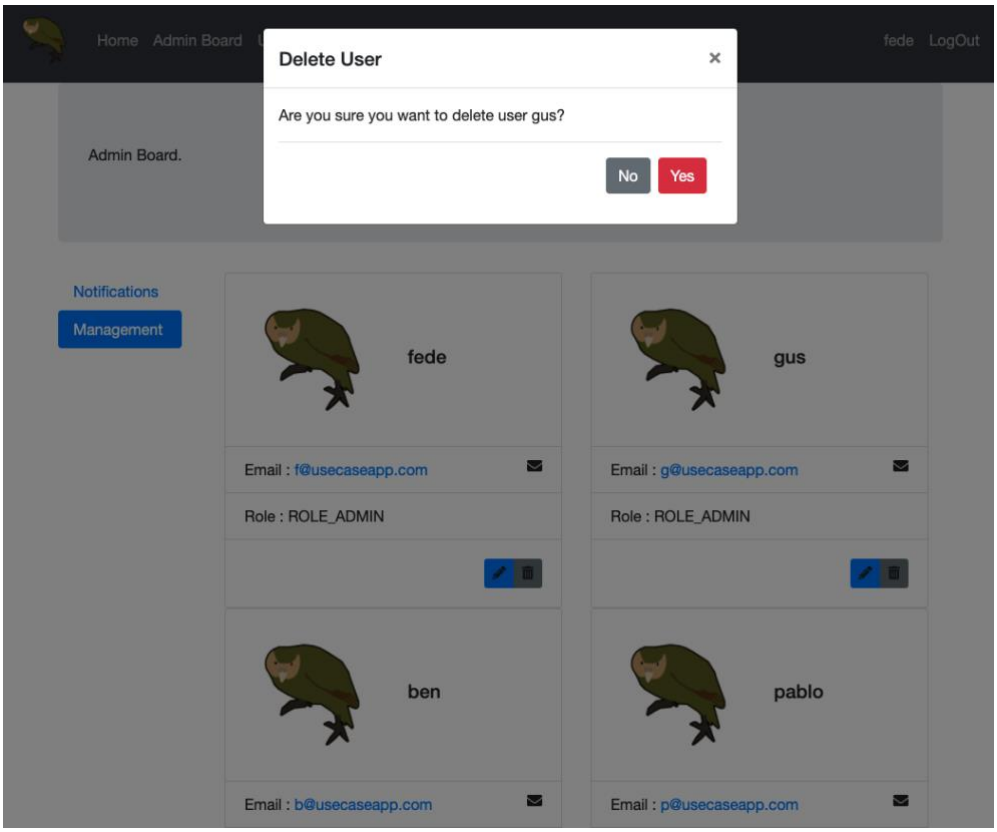

Figura 46: Manual: Web Administrador usuarios eliminar.

El botón de eliminar lanzará un modal con un diálogo para confirmar o cancelar la eliminación.

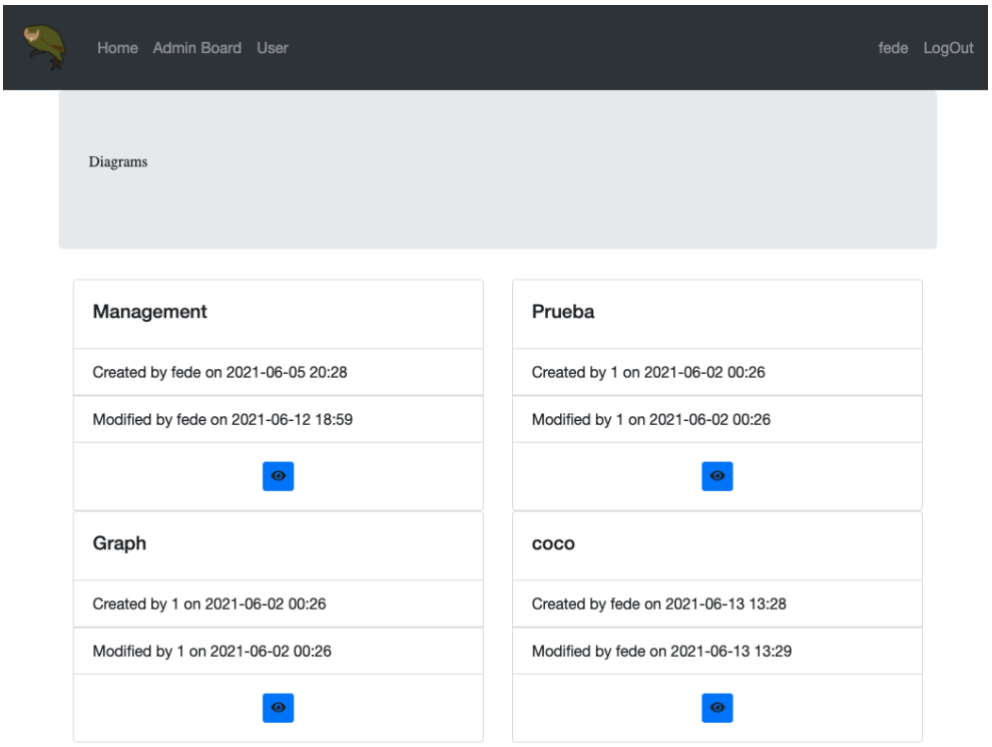

Figura 47: Manual: Web Usuario diagramas lista.

Tanto usuario como administrador tienen acceso al panel “Usuario”, donde aparecerá un listado con los diagramas asociados a ese usuario. La lista estará formada por tarjetas que contienen información sobre el diagrama. Mostrará el usuario que crea el diagrama junto con la fecha de la creación y el último usuario que ha realizado una modificación y la fecha de esta.

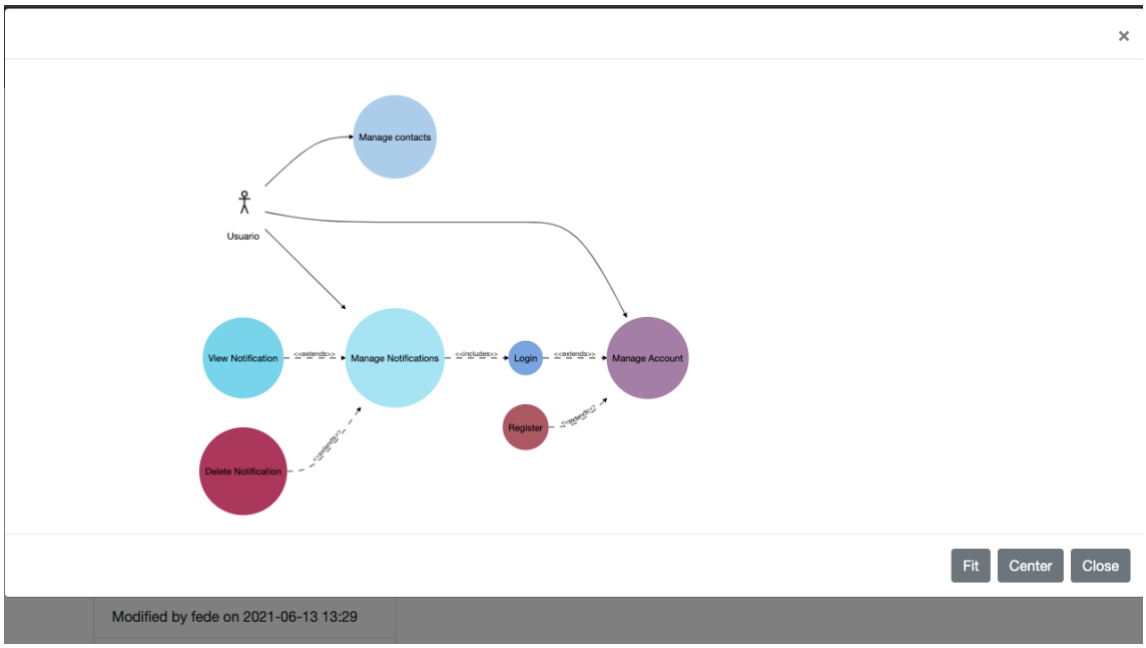

Figura 48: Manual: Web Usuario diagramas detalle.

Por último, al presionar el botón para visualizar el diagrama, se nos abrirá un modal donde se nos mostrará el diagrama. El *layout* permitirá aumentar y reducir el tamaño del diagrama, así como mover los elementos de sitio. También se han introducido unos botones para permitir ajustar el diagrama al tamaño del contendor y centrarlo.